

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-155845

(43)Date of publication of application : 06.06.2000

(51)Int.Cl.

G06T 11/00  
G06F 3/153  
G06F 12/00  
G06F 12/04  
G06T 1/00

(21)Application number : 11-140238

(71)Applicant : MITSUBISHI ELECTRONICS AMERICA  
INC

(22)Date of filing : 20.05.1999

(72)Inventor : ELIZABETH J SHURAPPU

(30)Priority

Priority number : 98 86554      Priority date : 21.05.1998      Priority country : US

98 164858 01.10.1998 US

98 164866	01.10.1998	US
-----------	------------	----

99 264261 08.03.1999 US

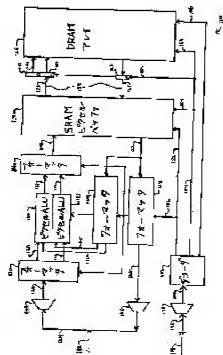
99 264281 08.03.1999 US

(54) STORAGE DEVICE, DATA FORMATTER, METHOD FOR ACCESSING DATA, METHOD FOR CLEARING AREA OF DATA, METHOD FOR COMPRESSING DATA, METHOD FOR FORMATTING DATA, AND GRAPHIC SYSTEM OPERATION METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To obtain rendering which is made fast by composing a storage device using a plurality of system of pixel arithmetic-logic units(ALUs) which are coupled with a pixel buffer, an input/output data formatter, and a read/write data formatter.

**SOLUTION:** Input graphics data are demultiplexed by an input data demultiplexer 126, and transferred to the input data formatter 130 through an input data bus 128 and formatted, and the data are transmitted to pixel ALUs 120 and 121 for processing through input data buses 129 and 131. The data are formatted by a formatter 140 from a pixel ALU to an SRAM and allocated to an SRAM pixel buffer 118 through a data bus 141. The SRAM pixel buffer 118 reads out the data formatted by the formatter 140 from the pixel ALU to the SRAM through a pixel ALU data bus 138.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Scanned 10/17/2007

Your Ref: 07844-499JP1  
Our Ref: PA1101

**Translation of Selected Portions of  
Pat. Laid-open Official Gazette**

-----  
Appln. No: 11-140238  
Appln. Date: May 20, 1999  
Laid-open Pub. No: 2000-155845  
Laid-open Pub. Date: June 6, 2000  
Priorities: 5/21/98 U.S.S.N. 60/086554, 10/1/98  
U.S.S.N. 09/164858, 10/1/98 U.S.S.N.  
09/164866, 3/8/99 U.S.S.N. 09/264261 &  
3/8/99 U.S.S.N. 09/264281

Inventor(s): Elizabeth J Shulapp (?)  
Applicant(s): Mitsubishi Electronics America Inc.  
Attorney(s): Kuro Fukami et al.  
-----

**1. Title of the Invention**

MEMORY DEVICE, DATA FORMATTER, A METHOD FOR ACCESSING  
DATA, A METHOD FOR CLEARING A DATA REGION, A METHOD  
FOR COMPRESSING DATA, A METHOD FOR FORMATTING DATA, A  
GRAPHIC SYSTEM AND A METHOD FOR OPERATING A GRAPHIC  
SYSTEM

**2. Claims**

(omitted)

**3. Detailed Description of the Invention (Selected Portions)**

1)

(omitted)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-155845

(P2000-155845A)

(43) 公開日 平成12年6月6日(2000.6.6)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テマート <sup>7</sup> (参考)
G 0 6 T 11/00		G 0 6 F 15/72	A
G 0 6 F 3/153	3 3 6	3/153	3 3 6 B
12/00	5 8 0	12/00	5 8 0
12/04	5 3 0	12/04	5 3 0
G 0 6 T 1/00		15/66	J

審査請求 未請求 請求項の数53 O L (全 77 頁)

- (21) 出願番号 特願平11-140238
- (22) 出願日 平成11年5月20日(1999.5.20)
- (31) 優先権主張番号 60/088554
- (32) 優先日 平成10年5月21日(1998.5.21)
- (33) 優先権主張国 米国 (US)
- (31) 優先権主張番号 09/164858
- (32) 優先日 平成10年10月1日(1998.10.1)
- (33) 優先権主張国 米国 (US)
- (31) 優先権主張番号 09/164866
- (32) 優先日 平成10年10月1日(1998.10.1)
- (33) 優先権主張国 米国 (US)

- (71) 出願人 591260649  
ミツビシ・エレクトロニクス・アメリカ・  
インコーポレーテッド  
アメリカ合衆国, 90630-0007 カリフォルニア州, サイプレス、プラザ・ドライ  
ブ、5665
- (72) 発明者 エリザベス・ジェイ・シュラップ  
アメリカ合衆国, 95132 カリフォルニア  
州, サン・ノゼ、ベクスレー・ランディ  
ング、1841
- (74) 代理人 100064746  
弁理士 深見 久郎 (外3名)

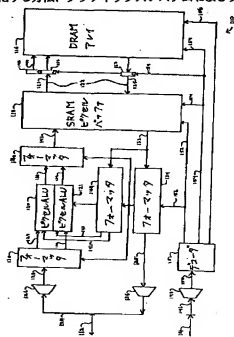
最終頁に続く

- (54) 【発明の名称】 記憶装置、データフォーマット、データにアクセスする方法、データの領域をクリアする方法、データを圧縮する方法、データをフォーマット化する方法、グラフィックスシステムおよびグラ

(57) 【要約】

【課題】 コンピュータグラフィックスシステムにおいて二次元および三次元イメージの増速されたレンダリングをもたらす半導体記憶装置を提供する。

【解決手段】 半導体記憶装置は、メモリアレイ (116) と、ピクセルバッファ (118) と、ピクセルバッファに結合される複数のピクセル算術演算装置 (120、121) と、入力データフォーマット (130) と、出力データフォーマット (134) と、読出データフォーマット、書込データフォーマットと、アドレスおよび制御入力バス (146) とを含む。



## 【特許請求の範囲】

【請求項1】 メモリアレイと、前記メモリアレイに結合されるバッファと、前記バッファに結合される複数の算術論理演算装置とを含む記憶装置であって、さらに前記複数の算術論理演算装置に結合され、かつ前記記憶装置の外部にあるレンダリングバスに結合される入力データフォーマットと、前記バッファおよび外部レンダリングバスに結合される出力データフォーマットと、前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマットと、前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマットと、前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマット、前記出力データフォーマット、前記書込データフォーマット、前記読出データフォーマットおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入力バスとを含む記憶装置。

【請求項2】 コントローラに結合されるメモリのためのデータフォーマットであって、前記コントローラから前記データフォーマットへ送信される1つ以上の信号から1組のピクセル差成分を抽出する第1のフォーマットと、抽出されたピクセル差成分と1組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータとを含む、データフォーマット。

【請求項3】 前記新しい組のピクセル成分をフォーマット化する第2のフォーマットを含む、請求項2に記載のデータフォーマット。

【請求項4】 レンダリングバスに結合される記憶装置のためのデータフォーマットであって、ピクセルデータから複数のピクセル成分を抽出する第1のフォーマットを含む、各抽出されたピクセル成分はピクセルに対するウィンドウ識別データフィールドに対応しており、前記データフォーマットはさらに前記第1のフォーマットに結合され、複数の選択可能なオペレーションのモードをストアするメモリユニットと、

前記メモリユニットに結合され、前記メモリユニットにストアされる1つ以上の選択可能なオペレーションのモードに従って新しいピクセルデータをフォーマット化する第2のフォーマットを含む、データフォーマット。

【請求項5】 ピクセルデータから複数のピクセル成分を抽出し、抽出されたピクセル成分を複数の群のピクセルデータに組合せる第1のフォーマットと、前記第1のフォーマットに結合され、前記複数の群のピクセルデータを組立てる第2のフォーマットとを含む、記憶装置のためのデータフォーマット。

【請求項6】 ピクセルデータから前記ピクセルデータの複数の群を抽出する第1のフォーマットと、前記第1のフォーマットに結合され、前記ピクセルデータの群から複数のピクセル成分を抽出する第2のフォーマットとを含む、記憶装置のためのデータフォーマット。

【請求項7】 記憶装置であってデータをストアするためのメモリアレイを含み、前記メモリアレイは複数のメモリバンクを含み、各メモリバンクは複数のメモリセルと、前記複数のメモリセルにアクセスするように結合される複数のセンスアンプとを含んでおり、前記記憶装置はさらに前記メモリアレイの前記センスアンプに結合されるバッファを含み、前記バッファは複数のキャッシュラインを含んでおり、前記記憶装置はさらに前記バッファに結合される複数の算術論理演算装置と、前記複数の算術論理演算装置に結合され、前記記憶装置の外部にあるレンダリングバスに結合される入力データフォーマットと、

前記バッファおよび外部レンダリングバスに結合される出力データフォーマットと、前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマットと、前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマットと、前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマット、前記出力データフォーマット、前記書込データフォーマット、前記読出データフォーマットおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入力バスとを含む、記憶装置。

【請求項8】 前記入力データフォーマットは、前記レンダリングバスを介して前記入力データフォーマットへ送信される1組のピクセル差成分を抽出する第1のフォーマットと、抽出されたピクセル差成分と1組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータとを含む、請求項7に記載の記憶装置。

【請求項9】 前記新しい組のピクセル成分をフォーマット化する第2のフォーマットを含む、請求項8に記載の記憶装置。

【請求項10】 メモリバンクオペレーションを特定するメモリバンクコマンドを受信するステップと、グローバルバスオペレーションを特定するグローバルバスコマンドを受信するステップと、算術論理演算装置オペレーションを特定するデータ処理コマンドを受信するステップと、前記メモリバンクコマンドに従って特定されたメモリバンクオペレーションを実行し、前記グローバルバスコマ

ンドに従って特定されたグローバルバスオペレーション  
を実行し、前記データ処理コマンドに従って特定された  
算術論理演算装置オペレーションを実行するステップと  
を含む、少なくとも1つの記憶装置内のデータにアクセ  
スする方法。

【請求項11】 メモリのキャッシュライン内にストア  
される複数のバイトのデータの各々に対応するデータを書  
込イネーブルするか、または書込ディスエーブルする  
かを制御するために少なくとも1つのバイトマスクレジ  
スタを設定するステップと、  
メモリの前記キャッシュライン内にストアされる前記複  
数のバイトのデータの各々の成分に対応するデータを書  
込イネーブルするか、または書込ディスエーブルする  
かを制御するために少なくとも1つのブレンマスクレジ  
スタを設定するステップと、  
前記少なくとも1つのバイトマスクレジスタおよび前記  
少なくとも1つのブレンマスクレジスタの書込イネー  
ブル設定により制御されるように、特定されたメモリバ  
ンクおよびコラムアドレスに結合される1組の前もって  
活性化されたセンスアンプにキャッシュラインメモリの  
内容を書込むステップとを含む、記憶装置内のデータの  
傾転をクリアする方法。

【請求項12】 キャッシュラインメモリの内容を書込  
む前記ステップを、特定された数のメモリバンクおよび  
コラムアドレスに結合される特定された数の組の前も  
って活性化されたセンスアンプに対して、特定された回数  
だけ繰返すステップを含む、請求項11に記載の方法。

【請求項13】 前記少なくとも1つのバイトマスクレ  
ジスタおよび前記少なくとも1つのブレンマスクレジ  
スタの前記書込イネーブル設定により制御されるよう  
に、前記キャッシュラインメモリの前記内容を書込む前  
記ステップは、特定されたメモリバンクおよびコラム  
アドレスに結合される複数の組の前もって活性化され  
たセンスアンプに対して同時に実行される、請求項11に  
記載の方法。

【請求項14】 コントローラからメモリへ送信される  
データを圧縮する方法であって、  
前記コントローラおよび前記メモリの両方に、複数の古  
いビクセルに対応する古いビクセルデータをストアする  
ステップを含み、前記古いビクセルデータは前記古いビ  
クセルデータにより表わされる古いビクセルの各々に対  
する1組の古いビクセル成分を含んでおり、前記方法は  
さらに前記コントローラにおいて、複数の新しいビクセ  
ルのための新しいビクセルデータを計算するステップ  
を含み、前記新しいビクセルデータは前記新しいビクセ  
ルデータにより表わされる新しいビクセルの各々に対  
する1組の新しいビクセル成分を含んでおり、前記方法はさ  
らに前記コントローラにおいて、前記新しいビクセル  
データのある特定のビクセル成分と前記古いビクセル  
データのある特定のビクセル成分との差を計算するステップ

と、  
前記コントローラにおいて、前記新しいビクセルデータ  
のある特定のビクセル成分と前記古いビクセルデータ  
のある特定のビクセル成分との前もって計算された差の差  
を計算するステップと、

前記コントローラにおいて、計算された差と、差の計算  
された差とに基づいて圧縮フォーマットを決定するステ  
ップと、

前記コントローラにおいて前記計算された差と前記差の  
計算された差とを決定された圧縮フォーマットにフォー  
マット化することにより前記新しいビクセルデータを圧  
縮するステップと、

圧縮された新しいビクセルデータを前記コントローラか  
ら前記メモリへ送信するステップと、

前記圧縮された新しいビクセルデータを前記メモリ内で  
伸長するステップとを含む、方法。

【請求項15】 前記コントローラにおいて複数の新し  
いビクセルのための新しいビクセルデータを計算する前  
記ステップは、差の値を、これらが計算された元のビク  
セルデータの幅に符号拡張するステップを含み、前記新  
しいビクセルデータは前記新しいビクセルデータにより  
表わされる新しいビクセルの各々に対する1組の新しい  
ビクセル成分を含む、請求項14に記載の方法。

【請求項16】 前記圧縮された新しいビクセルデータ  
を前記メモリ内で伸長する前記ステップは、  
複数のデータフォーマットから前記計算された差と前記  
差の計算された差とを抽出するステップと、  
抽出された差および差の差を、対応する古いビクセル成  
分に対応するデータ幅に符号拡張するステップと、

符号拡張された差および差の差をストアされた古いビ  
クセル成分に加算することによって新しいビクセル成分を  
再生するステップと、

前記新しいビクセルデータをフォーマット化するステ  
ップとを含む、請求項14に記載の方法。

【請求項17】 ビクセルデータを含むメモリのキャ  
ッシュラインから複数のウィンドウ識別ビクセル成分を抽  
出するステップと、

ウィンドウ識別フィールドが抽出された各ビクセルに対  
して、色ビクセルデータのどの部分とオーバーレイビクセ  
ルデータのどの部分とを前記コントローラに送信するか  
を決定するステップと、

抽出された複数のウィンドウ識別ビクセル成分、色ビ  
クセルデータの決定された部分およびオーバーレイビクセ  
ルデータの決定された部分を前記コントローラに送信する  
ステップとを含む、メモリからコントローラへ送信され  
るデータをフォーマット化する方法。

【請求項18】 少なくとも1つの算術論理演算装置か  
ら複数のビクセル成分を受信するステップと、  
前記複数のビクセル成分をビクセルデータの群にバック  
するステップと、

前記ピクセルデータの群を1ブロックのデータに組立てるステップとを含む、記憶装置においてデータをフォーマット化する方法。

【請求項19】 少なくとも1つの算術論理演算装置から複数のピクセル成分を受ける前記ステップはピクセル色成分を受信するステップと、ピクセルデプス成分を受信するステップと、ピクセルステンシル成分を受信するステップとを含む、請求項18に記載の方法。

【請求項20】 バッファからピクセルデータのブロックを受信するステップと、前記ピクセルデータのブロックから複数の群の選択されたピクセルデータを抽出するステップと、前記複数の群の選択されたピクセルデータから複数の選択されたピクセル成分を抽出するステップとを含む、記憶装置においてデータをフォーマット化する方法。

【請求項21】 コンピュータによる動作のためのグラフィックスシステムであって、レンダリングコントローラと、前記レンダリングコントローラと、前記グラフィックスシステムを動作させるのに用いられる前記コンピュータとに結合されるインタフェースと、記憶装置と、前記レンダリングコントローラおよび前記記憶装置に結合されるレンダリングバスと、前記レンダリングコントローラおよび前記記憶装置に結合されるアドレスおよび制御バスを含む、グラフィックスシステム。

【請求項22】 ビデオ出力チャネルにより前記レンダリングコントローラに結合されるビデオ出力回路を含む、請求項21に記載のグラフィックスシステム。

【請求項23】 前記記憶装置は、前記レンダリングバスと前記アドレスおよび制御バスとに結合される複数のメモリチップを含む、請求項21に記載のグラフィックスシステム。

【請求項24】 レンダリングバスとアドレスおよび制御バスとの複数の対を含む、請求項23に記載のグラフィックスシステム。

【請求項25】 最大数のメモリチップが、レンダリングバスとアドレスおよび制御バスとの各対に結合される、請求項24に記載のグラフィックスシステム。

【請求項26】 レンダリングバスとアドレスおよび制御バスとの各対に結合されるメモリチップの前記最大数は4である、請求項25に記載のグラフィックスシステム。

【請求項27】 前記レンダリングバスは同時双方向送受信を含み、データを同時に前記レンダリングコントローラから前記記憶装置に送信し前記記憶装置から前記レンダリングコントローラへ送信することを可能にする、請求項21に記載のグラフィックスシステム。

【請求項28】 前記記憶装置はデータをストアするためのメモリアレイと、前記メモリアレイに結合されるバッファと、前記バッファに結合され、前記記憶装置の外部にあるレンダリングバスに結合される複数の算術論理演算装置と、外部レンダリングバスおよび前記複数の算術論理演算装置に結合される入力データフォーマットと、前記バッファおよび前記外部レンダリングバスに結合される出力データフォーマットと、前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマットと、前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマットと、前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマット、前記出力データフォーマット、前記書込データフォーマット、前記読出データフォーマットおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入出力バスを含む、請求項21に記載のグラフィックスシステム。

【請求項29】 前記メモリアレイは複数のメモリアレイを含み、各メモリアレイは複数のメモリアレイを含み、各メモリアレイは複数のメモリアレイを含み、各メモリアレイは複数のメモリアレイを含み、各メモリアレイは1ビットのデータをストアすることができる、請求項28に記載のグラフィックスシステム。

【請求項30】 前記複数の算術論理演算装置は、複数のラスタオペレーションユニット、複数のブレンドユニット、複数のデプスユニット、複数のステンシルユニットおよび複数のウィンドウ識別ユニットを含み、複数のピクセルに対応するデータを同時に処理するようにする、請求項28に記載のグラフィックスシステム。

【請求項31】 前記入力データフォーマットは、第2のフォーマットに結合されるアキュムレータに結合される第1のフォーマットを含む、請求項28に記載のグラフィックスシステム。

【請求項32】 前記出力データフォーマットは、複数のウィンドウ識別データ抽出器と、複数のオーバーレイデータセレクトと、複数の色データセレクトとに結合されるモードレジスタを含む、請求項28に記載のグラフィックスシステム。

【請求項33】 前記読出データフォーマットはモードレジスタおよびマスキングレジスタを含み、前記レジスタの各々は複数のデータ抽出器および複数のデータアンバッファに結合される、請求項28に記載のグラフィックスシステム。

【請求項34】 前記書込データフォーマットはモードレジスタおよびマスキングレジスタを含み、前記レジスタの各々は複数の色バッファユニットと、複数のデプスバッ

ユニットと、複数のエクストラバックユニットとに結合される、請求項28に記載のグラフィックスシステム。

【請求項35】 前記アドレスおよび制御入力バスは、複数のアドレスおよび制御チャネルに結合されるデコーダを含む、請求項28に記載のグラフィックスシステム。

【請求項36】 コンピュータによる動作のためのグラフィックスシステムであって、コントローラに結合される記憶装置のためのデータフォーマットを含む、前記データフォーマットは前記コントローラから前記データフォーマットへ送信される1つ以上の信号から1組のピクセル差成分を抽出する第1のフォーマットと、

抽出されたピクセル差成分と1組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータを含む、グラフィックスシステム。

【請求項37】 前記新しい組のピクセル成分をフォーマット化する第2のフォーマットを含む、請求項36に記載のグラフィックスシステム。

【請求項38】 コンピュータによる動作のためのグラフィックスシステムであって、レンダリングバスに結合される記憶装置のためのデータフォーマットを含む、前記データフォーマットはピクセルデータから複数のピクセル成分を抽出する第1のフォーマットを含む、各抽出されたピクセル成分はピクセルに対するウィンドウ識別データフィールドに対応しており、前記データフォーマットはさらに前記第1のフォーマットに結合され、複数の選択可能なオペレーションのモードをストアするメモリユニットと、前記メモリユニットに結合され、前記メモリユニットにストアされる1つ以上の前記選択可能なオペレーションのモードに従って新しいピクセルデータをフォーマット化する第2のフォーマットを含む、グラフィックスシステム。

【請求項39】 コンピュータによる動作のためのグラフィックスシステムであって、記憶装置のためのデータフォーマットを含む、前記データフォーマットはピクセルデータから複数のピクセル成分を抽出して、抽出されたピクセル成分を複数の群のピクセルデータに組合せる第1のフォーマットと、前記第1のフォーマットに結合され、前記複数の群のピクセルデータを組立てる第2のフォーマットを含む、グラフィックスシステム。

【請求項40】 コンピュータによる動作のためのグラフィックスシステムであって、記憶装置のためのデータフォーマットを含む、前記データフォーマットはピクセルデータから前記ピクセルデータの複数の群を抽出する第1のフォーマットと、前記第1のフォーマットに結合され、前記ピクセルデータの群から複数のピクセル成分を抽出する第2のフ

ォーマットとを含む、グラフィックスシステム。

【請求項41】 コンピュータによる動作のためのグラフィックスシステムであって、データをストアするためのメモリアレイを含む、前記メモリアレイは複数のメモリバンクを含む、各メモリバンクは複数のメモリセルと、前記複数のメモリセルにアクセスするよう結合される複数のセンスアンプとを含んでおり、前記グラフィックスシステムはさらに前記メモリアレイの前記センスアンプに結合されるバッファを含み、前記バッファは複数のキャッシュラインを含み、前記グラフィックスシステムはさらにピクセルバッファに結合され、記憶装置の外部にあるレンダリングバスに結合される複数の算術論理演算装置と、外部レンダリングバスおよび前記複数の算術論理演算装置に結合される入力データフォーマットと、前記バッファおよび前記外部レンダリングバスに結合される出力データフォーマットと、前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマットと、前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマットと、前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマット、前記出力データフォーマット、前記書込データフォーマット、前記読出データフォーマットおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入力バスを含む、グラフィックスシステム。

【請求項42】 前記入力データフォーマットは前記レンダリングバスを介して前記入力データフォーマットへ送信される1組のピクセル差成分を抽出する第1のフォーマットと、抽出されたピクセル差成分と1組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータを含む、請求項41に記載のグラフィックスシステム。

【請求項43】 前記新しい組のピクセル成分をフォーマット化する第2のフォーマットを含む、請求項42に記載のグラフィックスシステム。

【請求項44】 コンピュータに関連してグラフィックスシステムを動作させる方法であって、レンダリングコントローラから記憶装置へレンダリングバスを介して入力データを送信するステップと、前記レンダリングコントローラから前記記憶装置へアドレスおよび制御バスを介してアドレスおよび制御データを送信するステップと、前記入力データをフォーマット化するステップと、フォーマット化された入力データを処理して処理されたデータを得るステップと、前記処理されたデータをフォーマット化するステップと、



フォーマット化された処理されたデータをメモリにストアするステップと、

前記処理されたデータを前記メモリからアクセスするステップと、

アクセスされた処理されたデータのいくつかをビデオ出力回路への送信のためにフォーマット化し、かつそのようにフォーマット化された処理されたデータを前記記憶装置から前記レンダリングバスを介して送信するステップと、

再処理すべき前記アクセスされた処理されたデータのいくつかをフォーマット化し、かつそのようにフォーマット化された前記再処理すべき処理されたデータを送信するステップを含む。方法。

【請求項 45】 前記レンダリングコントローラから前記記憶装置へアドレスおよび制御バスを介してアドレスおよび制御データを送信する前記ステップは、メモリバンクオペレーションを特定するメモリバンクコマンドを送信するステップと、グローバルバスオペレーションを特定するグローバルバスコマンドを送信するステップと、ピクセル算術論理演算装置オペレーションを特定するデータ処理コマンドを送信するステップを含む。フォーマット化された入力データを処理して処理されたデータを得る前記ステップと、フォーマット化された処理されたデータをメモリにストアする前記ステップと、前記処理されたデータを前記メモリからアクセスする前記ステップとは前記メモリバンクコマンドに従って特定されたメモリバンクオペレーションを実行し、前記グローバルバスコマンドに従って特定されたグローバルバスオペレーションを実行し、前記データ処理コマンドに従って特定されたピクセル算術論理演算装置オペレーションを実行するステップを含む。請求項 44 に記載の方法。

【請求項 46】 フォーマット化された処理されたデータをメモリにストアする前記ステップはメモリのキャッシュライン内にストアされる複数のバइटのデータの各々に対応するデータを書込イネーブルするか、または書込ディスエーブルするかを制御するために少なくとも 1 つのバ이트マスクレジスタを決定するステップと、メモリの前記キャッシュライン内にストアされる前記複数のバ이트のデータの各々の成分に対応するデータを書込イネーブルするか、または書込ディスエーブルするかどうかを制御するために少なくとも 1 つのプレーンマスクレジスタを設定するステップと、前記少なくとも 1 つのバ이트マスクレジスタおよび前記少なくとも 1 つのプレーンマスクレジスタの書込イネーブル設定により制御されるように、特定されたメモリバンクおよびコラムアドレスに結合される 1 組の前もって活性化されたセンスアンプにキャッシュラインメモリの内容を書込むステップを含む。請求項 44 に記載の方

法。

【請求項 47】 キャッシュラインメモリの内容を書込む前記ステップを、特定された数のメモリバンクおよびコラムアドレスに結合される特定された数の組の前もって活性化されたセンスアンプに対して、特定された回数だけ繰返すステップを含む。請求項 46 に記載の方法。

【請求項 48】 前記少なくとも 1 つのバ이트マスクレジスタおよび前記少なくとも 1 つのプレーンマスクレジスタの前記書込イネーブル設定により制御されるように、前記キャッシュラインメモリの前記内容を書込む前記ステップは、特定されたメモリバンクおよびコラムアドレスに結合される複数の組の前もって活性化されたセンスアンプに対して同時に実行される。請求項 46 に記載の方法。

【請求項 49】 レンダリングコントローラから記憶装置へレンダリングバスを介して入力データを送信する前記ステップと、前記入力データをフォーマット化する前記ステップとは、

前記レンダリングコントローラおよび前記メモリの両方に、複数の古いピクセルに対応する古いピクセルデータをストアするステップを含み、前記古いピクセルデータは前記古いピクセルデータにより表わされる古いピクセルの各々に対する 1 組の古いピクセル成分を含んでおり、前記ステップはさらに前記コントローラにおいて、複数の新しいピクセルのための新しいピクセルデータを計算するステップを含み、前記新しいピクセルデータは前記新しいピクセルデータにより表わされる新しいピクセルの各々に対する 1 組の新しいピクセル成分を含んでおり、前記ステップはさらに前記コントローラにおいて、前記新しいピクセルデータのある特定のピクセル成分と前記古いピクセルデータのある特定のピクセル成分との差を計算するステップと、前記コントローラにおいて、前記新しいピクセルデータのある特定のピクセル成分と前記古いピクセルデータのある特定のピクセル成分との前もって計算された差の差を計算するステップと、前記コントローラにおいて、計算された差と、差の計算された差とに基づいて圧縮フォーマットを決定するステップと、

前記コントローラにおいて前記計算された差と前記差の前記計算された差とを決定された圧縮フォーマットにフォーマット化することにより前記新しいピクセルデータを圧縮するステップと、圧縮された新しいピクセルデータを前記コントローラから前記メモリへ送信するステップと、前記圧縮された新しいピクセルデータを前記メモリ内で伸長するステップを含む。請求項 44 に記載の方法。

【請求項 50】 前記コントローラにおいて複数の新しいピクセルのための新しいピクセルデータを計算する前記ステップは、差の値を、これらが計算された元のデー

タの幅に符号拡張するステップを含み、前記新しいピクセルデータは前記新しいピクセルデータにより表わされる新しいピクセルの各々に対する1組の新しいピクセル成分を含む、請求項4 9に記載の方法。

【請求項5 1】 前記圧縮された新しいピクセルデータを前記メモリ内で伸張する前記ステップは、複数のデータフォーマットから前記計算された差と前記差の計算された差とを抽出するステップと、抽出された差および差の差を、対応する古いピクセル成分に対応するデータ幅に符号拡張するステップと、符号拡張された差および差の差をストアされた古いピクセル成分に加算することによって新しいピクセル成分を再生するステップと、

前記新しいピクセルデータをフォーマット化するステップとを含む、請求項4 9に記載の方法。

【請求項5 2】 アクセされたピクセルデータのいくつかをビデオ出力回路への送信のためにフォーマット化し、かつそのようにフォーマット化されたデータを前記レンダリングパスを介して送信する前記ステップは、ピクセルデータを含むメモリのキャッシュラインから複数のウィンドウ識別ピクセル成分を抽出するステップと、

ウィンドウ識別フィールドが抽出された各ピクセルに対して、色データのどの部分とオーバーレイデータのどの部分とを前記コントローラに送信するかを決定するステップと、

抽出された複数のウィンドウ識別ピクセル成分、色データの決定された部分およびオーバーレイデータの決定された部分を前記コントローラに送信するステップとを含む、請求項4 4に記載の方法。

【請求項5 3】 前記処理されたデータをフォーマット化する前記ステップは、少なくとも1つの算術論理演算装置から複数のピクセル成分を受信するステップと、

前記複数のピクセル成分をピクセルデータの群にパックするステップと、

前記ピクセルデータの群を1ブロックのピクセルデータに紐立てるステップとを含む、請求項4 4に記載の方法。

【発明の詳細な説明】

【000 1】

【発明の分野】 開示される発明は一般的にコンピュータシステムに関する。より特定的には、この発明はダイナミックランダムアクセスメモリ (DRAM) フレームバッファ装置と、その装置に基づく、増速された二次元および三次元グラフィックスレンダリングオペレーションを実行するためのアーキテクチャを提供するシステムとに関する。

【000 2】

【背景】 デュアルピクセル 3 DRAMチップおよびグラフィ

ックス処理システムは、高性能で高容量のフレームバッファを実現するのに用いられる。開示されるデュアルピクセル 3 DRAMチップおよびデュアルピクセル 3 DRAMチップに基づくグラフィックス処理システムのある局面は、1996年8月6日にデリーング (Deering) 他に発行された米国特許第5, 544, 306号に開示されており、この特許はここに完全に提示されているかのごとくにその全体においてこの開示に引用により援用される。

【000 3】 開示される発明は外部DRAMフレームバッファの使用に替るものを提示する。短期間の性能上の目的を満たすためには、組込みDRAMをフレームバッファメモリに用いたくなる。なぜなら、かなりの量の論理を実現するための表面積を残しつつ、ダイ上に4から8メガビットを組込むことが可能だろうからである。しかしながら、同じ時間フレームにおいて、グラフィックス指向の計算機プロダクトはフレームバッファメモリの10から80メガビットを要する。このため、グラフィックス処理計算機システムの要件を満たすようなフレームバッファを実現するためには、2個から10個の組込みDRAM装置が必要となるであろう。そのようなフレームバッファのフィル速度は非常に高速だろうが、計算機市場の大部分にとってはそのコストは高すぎるであろう。

【000 4】 二次元および三次元グラフィックイメージを処理する上で、フィル速度と一致させるためテクスチャマップを増速させなければならない。しかしながら、上述の分散されたフレームバッファではこれを効率よく行うことが難しくなる。テクスチャデータを分配する最も簡単なやり方は、各装置がすべてのものの複製を持つことだが、この方法は非常に効率の悪い組込みDRAMビットの用い方である。別のアプローチでは、データが複製されない態様でテクスチャデータをフレームバッファを含む装置の間で分配する。この選択肢では、組込みDRAMビットが有効に用いられるが、装置間で大量のテクスチャデータのルーティングも必要となる。

【000 5】 組込みDRAMを用いて単一の装置上でテクスチャキャッシュを実現する方がより実用的であるかもしれない。テクスチャデータはシステムメモリからページインしなければならないが、これはテクスチャデータが圧縮されている方がより効率よく働くであろう。

【000 6】 組込みDRAMは、セットアップユニットとラスタライザとの間の単純なFIFOを実現するのに用いることもできる。このFIFOにあり、大きな三角形がレンダリングされる間、ジオメトリおよびセットアップ処理を続けることが可能となる。これはまた、システムメモリからのテクスチャデータのページングおよび伸長のレイテンシを緩和するのに用いることもできる。

【000 7】 Talisman, Pixel FlowまたはOak's WARP5などの領域ベースのレンダリングアーキテクチャでは、レンダリングコントローラ上でフレームバッファの小さな部分をレンダリングし、次に最終の色値を外部DRAMに

転送する。コントローラはさらに次の領域をレンダリングし、さらにその次の領域へと、フレーム全体にわたってレンダリングし続ける。

【0008】隠面消去およびアンチエイリアシングに用いられる帯域幅のすべてがレンダリングコントローラに完全に残るため、フィル速度は外部帯域幅により制限されない。隠面消去およびアンチエイリアシングに用いられる記憶域のすべてはフレームバッファの小さい部分に対してのみ実現されるだけでよく、したがってレンダリングコントローラ上に置いておくことができる。

【0009】領域ベースのレンダリングの大きな欠点は、レンダリングを開始できるようになるまでに、各フレームごとにすべてのジオメトリを領域に分類してどこかにストアする必要があることである。この要件のため、一般的にフレームごとにレンダリングできるジオメトリの量に上限が課せられる。この制限は数多くの応用において許容できないものである。いくつかの領域ベースのレンダリングアーキテクチャは、大量のジオメトリを与えられた際にも外部DRAMから、および外部DRAMへ領域のためのデプス値および色値を書込むことによりある程度正確に機能できる。しかしながら、こうした実装では領域ベースのレンダリングの欠点のすべてが保たれたまま利点のすべてが損われる。

【0010】機械設計CADおよび他のコンテンツ作成応用ではジオメトリの複雑さに対する制限は許容できない。このような応用ではジオメトリの複雑さをフレーム速度とスムーズにトレードオフする能力が必要となる。このように、組込みDRAMのアプローチも領域ベースのレンダリングのアプローチも、現在のグラフィックス処理応用の性能の要求および実際のコスト制約を満たす十分な解決策を提供するものではない。

【0011】フレームバッファ設計に関する別の懸念は、シングルポートのフレームバッファメモリとデュアルポートのフレームバッファメモリとの性能トレードオフである。デュアルポートフレームバッファは専用ディスプレイポートを有しており、これによりレンダリングポートがより長い時間をレンダリングに費やすことができる。典型的に、ビデオランダムアクセスメモリ (VRAM) チップを含むデュアルポートフレームバッファは、そのビデオバッファがかなり大きいこと、そのフィル速度の約1%から2%しかビデオ転送オーバーレシジョンのために失わない、FBRAMチップ (3D-RAM) チップとも称される) を含むフレームバッファは、ビデオバッファがより小さいため、そのフィル速度の約5%から10%だけビデオ転送オーバーレシジョンのため失う。

【0012】シングルポートメモリは、ポートが高速双方向送受信をサポートするだけでなく、表示するためにピクセルデータを読出している際にはレンダリングすることができない。双方のレンダリングポート帯域幅が同じであるものとしてシングルポートメモリとデュアル

ポートメモリとを比較すると、デュアルポートメモリの方がフィル速度がより速く、コストもより高い。シングルポートの帯域幅がデュアルポートの帯域幅の合計に等しいものとしてシングルポートメモリとデュアルポートメモリとを比較すると、シングルポートメモリの方がより効率がよい。おおよそシングルポートメモリのフィル速度の方がデュアルポートメモリよりも高速であろう。すなわち、高帯域幅入力/出力(I/O)能力の出現により現在、帯域幅の制限が緩和されつつある限りにおいて、シングルポートメモリアーキテクチャの方がより効率のよいフレームバッファ性能を期待できる。

【0013】デュアルポートメモリではフレームバッファへのピクセルのフローがより滑らかである。シングルポートメモリはディスプレイデータのバーストを読出している間は定期的にレンダリングに利用できなくなる。レンダリングコントローラはシングルポートメモリとインターフェイスする際にはピクセルフローを滑らかにするのにより大きなピクセルFIFOを必要とする。より低コストのシステムでは、そのようなディスプレイバーストの間、レンダーラはアイドル状態となるかもしれない。

【0014】シングルポートメモリは、ダイ面積、ピン、パッケージング、テストおよび電力消費が小さいため費用がより安い。シングルポートメモリは同じ大きさのデュアルポートメモリと比べて、ビット当りの記憶コストがかなり低い。ビット当りのコストが低ければ、表示できないデータをフレームバッファにストアしてもそれほど問題にならない。

【0015】デュアルポートメモリはディスプレイ帯域幅が固定されている。必要となるディスプレイ帯域幅がより低ければ帯域幅は無駄になる。必要とされるディスプレイ帯域幅がより高ければ、そのメモリはそのディスプレイ要件に適していないことになる。シングルポートメモリには、レンダリング帯域幅とディスプレイ帯域幅とをトレードオフする柔軟性がある。緊急時には、シングルポートメモリは実際に非常に高いディスプレイ帯域幅を提供することができる。

【0016】デュアルポートメモリの専用ディスプレイポートは水平および垂直掃線消去期間の間は利用されないが、このことはディスプレイポートが約20パーセントの間はアイドル状態であることを意味する。

【0017】デュアルポートメモリでは、スクリーンへのピクセルおよびブロックのマッピングを固定することが要求される。比較して、シングルポートメモリではかなりの柔軟性を持ってピクセルおよびブロックをスクリーンへマッピングできる。

【0018】デュアルポートフレームバッファメモリは、レンダリングポートおよびディスプレイポートが異なるチップに接続されている場合には意味をなさない。両方のポートが同じチップに接続されている場合には、帯域幅の等しいシングルポートメモリの方が上に準

げた理由からより有効であろう。

【0019】シングルポートメモリでは、フレームバッファのビット当りの記憶コストがより安く、レンダリングチップとディスプレイチップとを単一の装置に併せさせることができるため、より低コストのシステムを製造できる。

【0020】このように、シングルポートメモリではデュアルポートメモリで設計できるものより低コストで低価格帯のシステムを設計することが可能になる。ビット当りの記憶コストはシングルポートメモリではかなり低いので高解像度/高ピクセル数設計において材料コストはかなり低くなる。シングルポートメモリはより柔軟であるため、より広範囲のプロダクト能力を提供する設計をもたらす。

# 【0021】

【発明の概要】この発明は、コンピュータグラフィックスシステムにおいて二次元および三次元イメージの増速されたレンダリングをもたらすシングルポートフレームバッファアクセスメモリ(デュアルピクセル3DRAM)チップに向けられる。

【0022】デュアルピクセル3DRAMチップは、レンダリングバスを介してレンダリングコントローラによりアクセスされるシングルポートの高速メモリを特徴とする。デュアルピクセル3DRAMチップは、DRAMレイ、SRAMピクセルバッファ、少なくとも1つのピクセル算術論理演算装置(ALU)およびグローバルバスを含む。また、デュアルピクセル3DRAMチップはいくつかのデータバスおよびデータフォーマットを含み、これらはグラフィックスデータがデュアルピクセル3DRAMチップ内で処理されたり、更新されたり、そのチップから送信されたり、そのチップ内にストアされたりする際にそのグラフィックスデータを割振り、フォーマット化する。

【0023】この発明の第1の局面において、デュアルピクセル3DRAMチップは、8ビットピクセルから最大512ビットピクセルまでの範囲のさまざまなピクセルサイズおよびフォーマットを処理するよう設定可能である。デュアルピクセル3DRAMチップはこれらの能力を実現するため、新規のプロトコルおよびデータバッキング方式を特徴とする。

【0024】この発明の別の局面において、デュアルピクセル3DRAMチップは、チップ上の2サイクルおよび3サイクルのピクセルALUオペレーションの両方を可能にする、レンダリングバスを介する可変入力および出力データレートをサポートする。

【0025】この発明の別の局面において、デュアルピクセル3DRAMチップは同時にオペレーションごとに2つの別個のピクセルまたはサンプルを処理する。

【0026】この発明の別の局面において、デュアルピクセル3DRAMチップは、チップとレンダリングコントローラとの間でより高速のフィル速度およびスループット

を可能にするデータ圧縮能力を特徴とする。

【0027】この発明の別の局面において、デュアルピクセル3DRAMチップは、アンチエイリアシングされたポリゴンをレンダリングするため新規のデルタ2アルゴリズムを用いるマルチサンプリング方式を用いる。

【0028】この発明の別の局面において、デュアルピクセル3DRAMチップは、レンダリングコントローラとチップとの間のアドレスおよび制御バス上の帯域幅要件を最小にするためチップ上にDRAMバンクおよびコラムアドレスを保持するための新規の方式を用いる。

【0029】この発明の別の局面において、デュアルピクセル3DRAMチップは、帯域幅のバランスを取りチップのオペレーション効率を最適化するような、チップ内部のデータバスの幅とデータ転送速度との関係を含む。

【0030】この発明の別の局面において、デュアルピクセル3DRAMチップは多倍精度ピクセルブレンドオペレーションを行ない、どんなビット幅の入力もブレンドされるようにする。

【0031】この発明の別の局面において、デュアルピクセル3DRAMチップはアドレスおよび制御情報を3つの別個の組の信号に分割し、3つの別個の組の信号は同時に送信されてチップ上でのDRAMバンクオペレーション、グローバルバスオペレーションおよびピクセルALUオペレーションを制御する。

【0032】この発明の別の局面において、デュアルピクセル3DRAMチップは、DRAMバンクコラムデコードとセンスアンプとの間の複数のバスに書き込み、この結果フレームバッファのクリア速度を4倍以上に増大させるFlash Line(フラッシュライン)オペレーションを特徴とする。

【0033】この発明の別の局面において、デュアルピクセル3DRAMチップは、DRAMレイとSRAMピクセルバッファとの間の双方向グローバルバスにより、異なるレベルのキャッシュ間でのデータの同時転送を可能にする、新規のオペレーションであるChange Cache Line(キャッシュライン変更)を特徴とする。

【0034】この発明の別の局面において、デュアルピクセル3DRAMチップは、ページリチャージオペレーションとページバンクアクセスオペレーションとを1つのオペレーションに組合せた、Change Page bank(ページバンク変更)オペレーションを特徴とする。

【0035】この発明の別の局面において、デュアルピクセル3DRAMチップは、ある特定のレジスタのデータまたは内容がチップ上のある特定のバスを介してブロードキャストされるピクセルALUオペレーションを特徴とする。

【0036】この発明の別の局面において、デュアルピクセル3DRAMチップは、SRAMピクセルバッファからシングルでも、デュアルピクセルフォーマットでも、ピクセルデータを読み出すための革新的な手段を特徴とする。

【0037】オペレーション、構造、部品の組立および組合せのさまざまな新規の詳細を含むこの発明の上述および他の特徴を以下に添付の図面に関連してより特定のに説明する。この発明の特定の実施例はここに例示としてのみ開示されており、クレームされる発明に限定を課するものではないことが理解されるであろう。この発明の原理および特徴はこの発明の範囲から逸脱することなく数多くのさまざまな実施例において用いられ得る。

#### 【0038】

【詳細な説明】以下に、現在発明者により企図されるこの発明を実施するためのベストモードに基づいてこの発明を詳細に説明する。以下、図面の簡単な説明において説明される添付の図面を参照するが、図面すべてを通じて要素に一貫した番号を振っている。この開示を通じて、当業者には既知のVerilogハードウェア記述言語（VHDL）構文法で表現されるデュアルピクセル3DRAMチップのさまざまな機能的な側面を記述する。

#### 【0039】目次

- 1.0 アーキテクチャ
- 1.1 ピクセルALU
  - 1.1.1 ROP/ブレンドユニット
  - 1.1.2 デプスユニット
  - 1.1.3 ステンシルユニット
  - 1.1.4 ウィンドウIDユニット
- 1.2 SRAMピクセルバッファ
- 1.3 メモリ構成
- 1.4 ピン構成
- 1.5 プロトコル
  - 1.5.1 DRAMバンクオペレーション
  - 1.5.2 グローバルバスオペレーション
  - 1.5.3 ピクセルALUオペレーション
- 1.6 オペレーションタイミング
- 1.7 レジスタ
  - 1.7.1 識別
  - 1.7.2 FeatureEnable
  - 1.7.3 PixelConfig
  - 1.7.4 StencilDepthConfig
  - 1.7.5 ColorOP[0]
  - 1.7.6 ColorOP[1]
  - 1.7.7 ConstantColor
  - 1.7.8 Byte Mask[1:0]
  - 1.7.9 Plane Mask[7:0]
  - 1.7.10 ColorWIDLUT[3:0]
  - 1.7.11 OverlayWIDLUT[3:0]
  - 1.7.12 DisplayConfig
- 1.8 高速領域クリア
- 2.0 データルーティング
- 2.1 入力データフォーマット
  - 2.1.1 ピクセル圧縮
  - 2.1.2 入力データフォーマット

- 2.1.3 アキュムレータ
- 2.1.4 最終フォーマット
- 2.2 出力データフォーマット
  - 2.2.1 RDAT, RPIX (8ビット、16ビット、32ビットピクセル) オペレーション
  - 2.2.2 RPIX (64ビットピクセル) オペレーション
  - 2.2.3 RPIX (96ビットピクセル) オペレーション
  - 2.2.4 RPIX (128ビットピクセル) オペレーション
- 2.3 SRAMからピクセルALUへのルーティング
  - 2.3.1 8ビット、16ビットおよび32ビットピクセルのSRAM構成
  - 2.3.2 64ビットピクセルのSRAM構成
  - 2.3.3 96ビットピクセルのSRAM構成
  - 2.3.4 128ビットピクセルのSRAM構成
  - 2.3.5 UnpackColors
  - 2.3.6 UnpackDepths
  - 2.3.7 UnpackExtras
  - 2.3.8 UnpackAlpha, UnpackRed, UnpackGreen, UnpackBlue
  - 2.3.9 UnpackDepth
  - 2.3.10 UnpackStencil
  - 2.3.11 UnpackWid
  - 2.3.12 SramToPaluData
- 2.4 ピクセルALUからSRAMへのデータルーティング
  - 2.4.1 PackColor
  - 2.4.2 PackDepth
  - 2.4.3 PackExtra
  - 2.4.4 PaluToSramData
- 2.5 ピクセルALUからSRAMへのマスク生成
  - 2.5.1 WriteEnableMask
  - 2.5.2 ピクセルアドレスマスク
  - 2.5.3 MaskDepth
  - 2.5.4 EnableMask
  - 2.5.5 SelectPlaneMask
  - 2.5.6 ピクセルALUからSRAMへのマスク
- 3.0 ピクセルフォーマット
  - 3.1 8ビットピクセルフォーマット
  - 3.2 16ビットピクセルフォーマット
  - 3.3 32ビットピクセルフォーマット
  - 3.4 64ビットピクセルフォーマット
  - 3.5 96ビットピクセルフォーマット
  - 3.6 128ビットピクセルフォーマット
  - 3.7 マルチ・サンプル・ポリゴン・アンチエイリアシング
    - 3.7.1 累算バッファ
    - 3.7.2 Aバッファ
    - 3.7.3 マルチサンプル
    - 3.7.3.1 サンプルあたり色のみ
    - 3.7.3.2 サンプルあたり色およびデプス

3.7.4 サンプルあたり色およびデプスの速度改良  
3.8 256ビットピクセルフォーマット(4×マルチサンプル)

3.9 512ビットピクセルフォーマット(6×マルチサンプル)

4.0 双方向I/O

1.0 アーキテクチャ

図1は、計算機システムにおいて動作するグラフィックスサブシステム100を図示する。グラフィックスサブシステム100は、ビデオディスプレイフレームバッファとも呼ばれる。グラフィックスサブシステム100は、レンダリングコントローラ102、その中でグラフィックスサブシステム100が動作する計算機システムへのインタフェース104、ビデオ出力回路106、レンダリングコントローラ102からビデオ出力回路106へ延びるビデオ出力チャネル108、ここに開示する1つまたは2つ以上のデュアルピクセル3DRAMチップ110、1つまたは2つ以上のレンダリングバス112、および1つまたは2つ以上のアドレスおよび制御バス114を含む。図1に示すように、ビデオ出力回路106はレンダリングコントローラ102と物理的に分離されているが、代替の実施例は、単一のチップまたは装置の中にレンダリングコントローラ102およびビデオ出力回路106の両方を含む。

【0040】図1に示すグラフィックスサブシステム100は、レンダリングバス112とアドレスおよび制御バス114との2つの別個の対に接続される4つのデュアルピクセル3DRAMチップの組110a~dおよび110e~hの2組を含む。ここでの構成では、レンダリングバス112とアドレスおよび制御バス114との対に接続できるデュアルピクセル3DRAMチップ110の最大数は4つである。しかし、図1の省略記号(...)によって示されるように、グラフィックスサブシステムにおいて使用されるレンダリングコントローラ102とデュアルピクセル3DRAMチップ110との間のレンダリングバスと制御バスとの対の数に制限はない。グラフィックスサブシステム100は、処理されるピクセルデータに依存して、シングル・バッファまたはダブル・バッファのいずれかであり、色バッファAおよびBならびに単一のZバッファを含む。グラフィックスサブシステム100は、8ビット/ピクセルから最大512ビット/ピクセルまでの範囲のさまざまな異なるピクセルフォーマットをサポートする能力を特徴とし、これによって、多数のフレームバッファ100の法に対応する。

【0041】レンダリングコントローラ102は、アドレスおよび制御バス114を通じてデュアルピクセル3DRAMチップ110a~hのための制御情報を伝送する。レンダリングコントローラ102は、レンダリングバス112を通じてデュアルピクセル3DRAMチップ110へのおよびデュアルピクセル3DRAMチップ110からのピ

クセルデータアクセスを行なう。レンダリングコントローラ102は、ピクセルアクセスのシーケンスをレンダリングオペレーションのシーケンスへ変換する。レンダリングバス112は、近年グラフィックス処理分野に出現したRDRAM<sup>®</sup>およびSLDRAMなどの高帯域通信アーキテクチャをサポートする。

【0042】レンダリングコントローラ102は、レンダリングバス112を通じてピクセルデータをデュアルピクセル3DRAMチップ110a~hに書き込み、デュアルピクセル3DRAMチップ110a~hは、レンダリングバス112を通じてレンダリングコントローラ102へ更新されたピクセルデータを転送する。レンダリングコントローラ102は、別個のアドレスおよび制御バス114を通じてフレームバッファ制御信号およびフレームバッファコマンドをデュアルピクセル3DRAMチップ110a~hへ転送する。フレームバッファコマンドおよびフレームバッファ制御信号は、デュアルピクセル3DRAMチップ110a~hの内部オペレーションを調整する。

【0043】デュアルピクセル3DRAMチップ110は、ビンの数を最小限に留めつつ、レンダリングコントローラ102とデュアルピクセル3DRAMチップ110a~hとの間で、制御情報を受信し、グラフィックスデータを送受信するため、高帯域入/出力(I/O)技術をサポートする。実施例については、レンダリングバス102は、データおよび制御I/O用のRambus Direct RDRAM<sup>®</sup>仕様に準拠し、1.8ギガビット/秒でピクセルデータを転送する18ビット双方向データバスと800メガビット/秒で情報を転送する単方向アドレスおよび制御バスとを有する。他実施例については、レンダリングバス102は、公式にはSyncLinkとして知られるオープンIEEEおよびJEDEC規格、SLDRAMに準拠する。以下に挙げる刊行物は、このような高帯域I/Oアーキテクチャを詳細に説明しており、その全体としてここに引用により援用される。援用される文献は、ピーター・ギリングハム(Peter Gillingham)による『SLDRAMのアーキテクチャおよび機能の概要』(“SLDRAM Architectural and Functional Overview”)、MOSAIDテクノロジーズ(MOSAID Technologies, Inc.)、1997年8月29日; IEEEコンピュータ学会マイクロプロセスおよびマイクロコンピュータ規格小委員会後援の『高速メモリインタフェース(SyncLink)規格草案』(“Draft Standard For A High-Speed Memory Interface (SyncLink)”)、草案(Draft) 0.99 IEEE p1596.7-1999X、1996年; 『400 Mb/s/ピンSLDRAM』4×118SLDRAMパイプライン方式8バンク、2.5V動作(“400Mb/s/pin SLDRAM 4M×18 SLDRAM pipelined, eightbank, 2.5V operation”)、Draft/Advance SLD4M18DR400 4 MEG×18 SLDRAM、SLDRAM協会発行、1997年9月22日である。

もちろん、異なるI/Oアーキテクチャをサポートするためデュアルピクセル3DRAMチップ110の他実施例も

可能である。

【0044】図2は、一実施例のデュアルピクセル3DRAMチップ110を示す。デュアルピクセル3DRAMチップ110は、図1のデュアルピクセル3DRAMチップ110a～hの各々々々実質的に同様である。デュアルピクセル3DRAMチップ110は、DRAMアレイ116、SRAMピクセルバッファ118、2つのピクセル算術論理演算装置(ALU)120および121、ならびに図示される実施例においては別個のグローバル書込バス123およびグローバル読出バス125それぞれを含むグローバルバス122を含む。デュアルピクセル3DRAMチップ110はまた、グラフィックスデータが処理されデュアルピクセル3DRAMチップ110内にストアされるのに伴い、グラフィックスデータを割振りフォーマット化するデータフォーマッタといくつかのデータバスとを含む。

【0045】グラフィックスデータは、レンダリングバス112を通じてデュアルピクセル3DRAMチップ110へ入りかつそこから出る。デュアルピクセル3DRAMチップ110へ送信されるグラフィックスデータは、入出力バス(I/Oバス)124により受信される。入力グラフィックスデータは、入力データデマルチプレクサ126によりデマルチプレクスされ、入力データバス128を通じて入力データフォーマッタ130へ転送され、そこでデータはフォーマット化され、次にフォーマット化されたデータは、それぞれ処理のためにピクセルALU120および121へと入力データバス129および131を通じて送信される。

【0046】デュアルピクセル3DRAMチップ110からレンダリングバス112へ送信されるグラフィックスデータは、SRAM出力データバス132を通じてSRAMピクセルバッファ118から送られる。レンダリングバス112上で受信されるまでに、グラフィックスデータは出力データフォーマッタ134によりフォーマット化され、出力データバス135を通じて送信され、出力データデマルチプレクサ136によりマルチプレクスされ、チップ110からI/Oバス124を通じて転送される。

【0047】データは、2つの別個のデータバス138および139を通じて、ピクセルALU120および121とSRAMピクセルバッファ118との間で割振られる。ピクセルALU120および121からSRAMピクセルバッファ118へと送信されるグラフィックスデータは、その途中、ピクセルALUからSRAMへのフォーマッタ140によりフォーマット化され、次にデータバス141を通じてSRAMピクセルバッファ118へと割振られる。

【0048】SRAMピクセルバッファ118からピクセルALU120および121へと送信されるグラフィックスデータは、SRAM出力データバス132を通じてSRAMからピクセルALUへのフォーマッタ144へ割振られ、そしてフォーマット化されたデータはSRAMピクセルバッファデータバス142および143を通じてピクセルALU1

20および121へ送信される。

【0049】チップ110上で行なわれるオペレーションを方向づけるために使用されるアドレスおよび制御情報は、アドレスおよび制御バス114を通じてデュアルピクセル3DRAMチップ110へ送信される。情報は、アドレスおよび制御入力バス146で受信され、アドレスおよび制御デマルチプレクサ148によりデマルチプレクスされ、アドレスおよび制御バス150に沿ってデコーダ151へと送信される。デコーダ151は、デマルチプレクスされたアドレスおよび制御情報を受信し、これをデコードし、次にデコードされた情報はピクセルALUオペレーションチャネル152を通じてピクセルALU120および121ならびにSRAMピクセルバッファ118へ送信され、グローバルバスオペレーションチャネル154を通じて、SRAMピクセルバッファ118、パイプラインレジスタ127および137ならびにDRAMアレイ116へと送信され、そして、バンクオペレーションチャネル156を通じてDRAMアレイ116へ送信される。

【0050】図3は、RAMBUS<sup>®</sup>または(以前にはSynchLinkとして知られていた)SLDRAM入力/出力インタフェース仕様のいずれかで動作するよう構成されるデュアルピクセル3DRAMチップ110の一実施例のダイサイズのプロアプランを示す。図2の機能ブロック図とは異なり、図3のデュアルピクセル3DRAMチップ110のレイアウトは、図2に示す機能的要素のいくつかがデュアルピクセル3DRAMチップ110の特定の実施例においてどのように物理的に実現され得るのかを示す。

【0051】たとえば、デュアルピクセル3DRAMチップ110のこの物理的実現例は、チップ110の4つの角に位置する4つのセクション116a～dへ機能的に分離されたDRAMアレイ116を特徴とする。この物理的な分離にもかかわらず、DRAMアレイ116は図2に示すように1つの機能的単位として動作する。図3のDRAMアレイ116は、図3の4つのセクション116a～dの各々の中のア～Hとラベリングされる8つのインターリーブされたモジュラーDRAMバンク158を含む。DRAMアレイ116と同様、8つのDRAMバンク158A～Hはチップ110の4つの角に物理的に配置されるが、これらは、(32個ではなく)8個の機能単位として動作する。

【0052】デュアルピクセル3DRAMチップ110の全体としてのDRAM容量は、特定のチップ110の構成において使用されるモジュラーDRAMバンク158の数に依存して幅がある。各DRAMバンク158は、センス増幅器160を含む1組のラインバッファを含む。図22および図23を参照されたい。(「ラインバッファ」および「センス増幅器」または「センスアンプ」という語は、ここでは交換可能なものとして使用され、いずれも要素160を指す。)各DRAMバンク158は、DRAMビットのいくつかのライン164を含む複数のDRAMページ162

を含む。

【0053】デュアルビクセル3DRAMチップ110の一実施例は、10個のインターリーブされたDRAMバンク158を含む全部で40メガビットのDRAMアレイ116を有し、各バンク158は512個のページ162を含み、各ページ162は8個の1024ビットライン164を含む。この実施例においては、各DRAMバンク158の構造は一定して4メガビットDRAMである（1024ビット/ライン\*8ライン/ページ\*512ページ/バンク=4194304ビット/バンク≒4メガビット/バンク）。デュアルビクセル3DRAMチップ110内のインターリーブされたDRAMバンク158の数を変えることによって、チップの内部アーキテクチャを変更することなくチップ110の総記憶容量を調整することができる。

【0054】ラインバッファ160はセンサ増幅器を含み、DRAMバンク158内にストアされたビクセルデータにアクセスするとき、キャッシュラインの第2のレベルとして働く。（キャッシュラインの第1のレベルは、SRAMビクセルバッファ118内のメモリのラインである。）ラインバッファ160はDRAMバンク158へと直接マッピングされる。一実施例においては、各ラインバッファ160が、対応するDRAMバンク158のページの1つをマッピングする。一実施例においては、ラインバッファのエントリは1024ビットのライン一つを含む。

【0055】再び図2を参照し、ビクセルバッファ118は高速マルチポートスタティックRAM（SRAM）構成要素である。データは、グローバルバス122を通じて、SRAMビクセルバッファ118とDRAMアレイ116との間を転送される。図示される実施例においては、グローバルバス122は2つの単方向バス、グローバル書込バス123とグローバル読出バス125を含む。

【0056】SRAMビクセルバッファ118は、ビクセルALUデータバス138を通じてビクセルALUからSRAMへのフォーマット140によりフォーマット化されたデータを、読出す。SRAMビクセルバッファ118は、SRAM出力データバス132を通じて、出力データフォーマット134およびSRAMからビクセルALUへのフォーマット144の両方にデータを書込む。出力データフォーマット134は、SRAMビクセルバッファ118からデータフィールドをアンパックし、レンダリングバス112を通じて送信されるディスプレイ出力用にフィールドのいくつかを再パックする。SRAMからビクセルALUへのフォーマット144はまた、ビクセルALU120および121が使用するようにデータフィールドをアンパックする。

【0057】一実施例においては、グローバル書込バス123およびグローバル読出バス125は各々、SRAMビクセルバッファ118とDRAMアレイ116との間で1024ビットを搬送し、一方、ビクセルALUデータバス138および139ならびにSRAM出力データバス132は

各々256ビット幅である。

【0058】一実施例においては、SRAMビクセルバッファ118は、8本のキャッシュラインを有し、各キャッシュラインはメモリ1024ビット（1キロビット）を含む。8キロビットSRAMビクセルバッファ118は8つの1キロビットキャッシュラインに編成される。他実施例においては、SRAMビクセルバッファ118は16本の1キロビットキャッシュラインに編成される。

【0059】グローバルバス122は、SRAMビクセルバッファ118とDRAMアレイ116のセンサアンプ160との間での通信を可能にする。好ましい実施例においては、グローバルバス122は1024ビット10ナノ秒デュアルバス123および125を含む。グローバル読出バス125は、読出パイプラインレジスタ127を通じてセンサアンプ160からSRAMビクセルバッファ118へデータを転送し、グローバル書込バス123は、データ書込パイプラインレジスタ137を通じてSRAMビクセルバッファ118からセンサアンプ160へビクセルデータおよびマスクデータを転送する。この実施例はまた、DRAMアレイ116内のどのビットに上書きするかを制御するため、マスク書込パイプラインレジスタ145およびマスク書込バス147を用いる。他実施例においては、グローバルバス122は、グローバル読出バス125およびグローバル書込バス123の両方を含むが、パイプラインレジスタ127、137および145は用いられない。さらに他の実施例においては、グローバルバス122は、SRAMビクセルバッファ118からの読出およびSRAMビクセルバッファ118への書込の両方のために交互に使用される単一の双方向バスを含む。

【0060】ビクセルALU120および121とSRAMビクセルバッファ118との間のデータ転送は、グローバルバス122を通じてのSRAMビクセルバッファ118とDRAMアレイ116との間のデータ転送とは異なる。一実施例においては、ビクセルALU120および121は、256ビット5ナノ秒のデータバス138および139を通じてデータを書込み、ビクセルALU120および121は256ビット5ナノ秒のバス142および143を通じて送信されるデータを読出す。

【0061】1.1. ビクセルALU  
ビクセルALU120および121は、SRAMビクセルバッファ118へのパイプライン方式でのリード・モディファイ・ライトオペレーションを可能にする。パイプライン方式のリード・モディファイ・ライトオペレーションは、Z-バッファ比較、RGBアラスタオペレーション、およびブレンドオペレーションを含む。好ましい実施例のSRAMビクセルバッファ118のマルチポート性により、グローバルバス122を通じてのDRAMアレイ116のラインバッファ160とSRAMビクセルバッファ118との間での全キャッシュラインの並列転送が可能になる。



【0062】図2および図3に図示するように、デュアルピクセル3DRAMチップ110は、オフチップ帯域幅要求を最小限にするための2つのオンチップピクセルALU120および121を特徴とする。96ビットまたは128ビットのピクセルを更新するためには、40ビットの色情報と32ビットのデプス情報の送信が必要である。

【0063】デュアルピクセル3DRAMグラフィックスサブシステム100は、チップ110上のピンの数を最小限にしつつ、レンダリングコントローラ102とデュアルピクセル3DRAMチップ110との間でデータおよび制御情報を送信するため、Direct RDRAM™ BY RAMBUSまたは（従前にはSyncLinkとして知られていた）SLDRAMなどの高帯域I/O技術を用いる。一実施例においては、デュアルピクセル3DRAMグラフィックスサブシステム100は、1.8ギガビット/秒で遷移する（すなわち立ち上がり端および立下がり端の両方におけるデータの遷移が400MHzクロックである）18ビット半二重方向データバス112と、および、800メガビット/秒で遷移するレンダリングコントローラ102からデュアルピクセル3DRAMチップ110a-hへの8ビット単方向制御バス114を使用する。この実施例においては、ピクセルALU120および121は、処理されるピクセルのフォーマットに依存して、200MHzまたは133MHzのいずれかで動作し、そのため、デュアルピクセル3DRAMチップ110のピンにおいて受信される狭帯域高速データストリームは、内部では4倍から6倍幅が広いデータストリームへとデマルチプレクスされる必要がある。同様に、デュアルピクセル3DRAMチップ110において処理されるデータは、レンダリングバス112を通じてレンダリングコントローラ102へ送られる前にマルチプレクスされなければならない。

【0064】図3を参照し、ピクセルALU120および121は、デュアルピクセル3DRAMチップ110の中央に位置づけられる。ピクセルALU120および121は、処理要素の2つの完全な組を含み、したがって、多くの状況下でピクセルALU120および121がオペレーションごとに2つのピクセルを処理することを可能にする。処理されるピクセルのフォーマットが、ピクセルALU120および121が2つのピクセルを個別に処理できるか、または、一度に単一のピクセルを処理するようそれらのリソースを組合せなければならないかどうかを決定する。処理要素の完全な組ひとつは、4つのラストオペレーション（ROP）/ブレンドユニット166、デプスユニット168、ステンシルユニット170およびウィンドウ識別（WID）ユニット172を含む。

【0065】図4は、処理要素を2組含むピクセルALU120および121の一実施例を示す。処理要素の第1の組は、第1のピクセルALU120に対応し、「0」とラベリングされている。処理要素の第2の組は、第2の

ピクセルALU121に対応し、「1」とラベリングされている。ROP/ブレンドユニット166は、ユニット166を実現するため使用される回路のいくつかが共有されることを示すため図では重なり合うように示されている。

【0066】図5はやはり処理要素の完全な組2つを含むピクセルALU120および121の実施例である。ROP/ブレンドユニット166は、別個のROPユニット174とブレンドユニット176とに分割される。この実施例では、ROP/ブレンドユニット166の間で共有される回路はない。この実施例は、4つの8ビットブレンドユニット178と4つの10ビットブレンドユニット180とを特徴とする。より大きなピクセルフォーマットに対応するため、容量の異なるブレンドユニットが設けられる。好ましい実施例では、ROP/ブレンドユニットは各々10ビットユニットである。

【0067】図6は、ピクセルALU120および121のいずれか一方の中の処理ユニットの完全な組を示す。チップ110上に組合せられる、ピクセルALU120および121はこれらユニットの完全な組2つを有し、したがって、デュアルピクセル3DRAMチップ110が多くの状況下でオペレーションごとに2つのピクセルを処理することができる。文字「S」は、レンダリングバス112を通じてピクセルALU120または121へ送信されるソースデータを示す。文字「D」は、グローバルバス112を通じてSRAMピクセルバッファ118からピクセルALU120または121へ送信される先行データを示す。文字「R」は、グローバルバス112を通じてSRAMピクセルバッファ118へ再び送信される結果データを示す。文字「DT」、「ST」および「WT」は、それぞれデプスユニット168、ステンシルユニット170およびウィンドウIDユニット172により行なわれるテストの結果である。処理ユニットの完全な組は、各ピクセルのアルファ成分、赤成分、緑成分および青成分の処理のための4つのROP/ブレンドユニット166を含む。デュアルピクセル3DRAMチップ110のこの実施例においては、ROP/ブレンドユニット166へのソースデータ入力ストリームおよび先行データ入力ストリームは各々11ビット幅である。ROP/ブレンドユニット166から出力される結果データストリームは、10ビット幅である。デプスユニット168については、ソースデータストリーム、先行データストリーム、および結果データストリームは32ビット幅である。ステンシルユニット170は、先行ストリームを受信して、結果ストリームを出力し、その両方とも8ビット幅である。ウィンドウIDユニット172は8ビットの先行データストリームを受信する。

【0068】1.1.1 ROP/ブレンドユニット  
図7は、一実施例における1つのROP/ブレンドユニット166をブロック図の形で示す。この実施例において

は、8個の10ビットROP/ブレンドユニット166が2つのピクセルALU120および121の中に位置する。8個のROP/ブレンドユニット166の各々は、機能的に同一であり、(ソース色(Sc)、ソース係数(Sf)、行先色(Dc)および行先係数(Df)用の)4つの11ビットデータ入力と結果用の10ビットデータ出力とを有する。各ROP/ブレンドユニット166は、(1)ROP(Sc、Pc、Dc)、(2)min(Sc、Dc)、(3)max(Sc、Dc)、(4)Sc\*Sf+Dc\*Df、(5)Sc\*Sf-Dc\*Df、または(6)Dc\*Df-Sc\*Sfの6つのオペレーションのうち1つを行なう。

【0069】ROPオペレーション、minオペレーション、maxオペレーションまたは8ビットブレンドオペレーションを行なうとき、8個のROP/ブレンドユニット166\*

\*は、すべて並列に作業できる。10ビットブレンドオペレーションを行なうときは、必要とされる処理を行なうためにブレンドユニット176の対を互いに組合せる必要がある。したがってデュアルピクセル3DRAMチップ110により10ビットブレンドオペレーションが行なわれるときには、一度に1つのピクセルしか処理できない。他実施例は、8個の10ビットROP/ブレンドユニット166を有し、いかなる場合にも一度に2つのピクセルを処理できる。

【0070】ソースブレンド係数「Sf」および行先ブレンド係数「Df」は、次の表1に示すようにソース色、行先色およびパターン色から導出される。

【0071】

【表1】

パラメータ	7bit	係数	Sf	Df
GL_ZERO	0		x	x
GL_ONE	1		x	x
GL_SRC_COLOR	Sc	Sf	SG	SB
GL_ONE_MINUS_SRC_COLOR	1-Sc	1-Sf	1-SG	1-SB
GL_DST_COLOR	Dc	Df	DG	DB
GL_ONE_MINUS_DST_COLOR	1-Dc	1-Df	1-DG	1-DB
GL_SRC_ALPHA	Sa		x	x
GL_ONE_MINUS_SRC_ALPHA	1-Sa		x	x
GL_DST_ALPHA	Da		x	x
GL_ONE_MINUS_DST_ALPHA	1-Da		x	x
GL_SRC_ALPHA_SATURATE	1	min(Sa, 1-Da)	x	
CL_CONSTANT_COLOR_EXT	Pa	Pf	Pg	Pb
CL_ONE_MINUS_CONSTANT_COLOR_EXT	1-Pa	1-Pf	1-Pg	1-Pb
CL_CONSTANT_ALPHA_EXT	Pa		x	x
CL_ONE_MINUS_CONSTANT_ALPHA_EXT	1-Pa		x	x

【0072】次に図8を参照し、ROP/ブレンドユニット166のラストオペレーション(ROP)部174が、256個のブルー演算のうち1つをソースSc入力、行先Dc入力およびパターンPc入力に対して行なう。アドレスおよび制御バス114を通じて送信される情報によりセットされるROPレジスタ182が、256個のブルー演算のうちどれを行なうかを決定する。3つの入力が必要と※

※される場合には、ラストオペレーションが行なわれる前に入力ひとつがパターンレジスタ184に書込まれる。ROP部174の1ビットスライスは、次のブルー方程式によって実現できる。

【0073】

【数1】

$$\begin{aligned}
 \text{Result} = & (\text{Op}[0] \& \sim \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) \mid (\text{Op}[1] \& \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) \mid \\
 & (\text{Op}[2] \& \sim \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) \mid (\text{Op}[3] \& \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) \mid \\
 & (\text{Op}[4] \& \sim \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) \mid (\text{Op}[5] \& \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) \mid \\
 & (\text{Op}[6] \& \sim \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) \mid (\text{Op}[7] \& \text{Dc} \& \sim \text{Sc} \& \sim \text{Pc}) ;
 \end{aligned}$$

【0074】図9から図14を参照し、ROP/ブレンドユニット166のブレンド部176は、処理回路または要素素なわち、1つのディザ計装装置186、2つの乗算

器188および190、加算器192、1つの切捨て装置194および1つのクランプ装置196を含む。図9に、8ビットブレンドユニット178を示す。オペレー

ジョンごとに2つのピクセル（またはアンチエイリアシングを行なうときにはサンプル）をブレンドできるようにするためにこれらのユニットが8つ必要である。

【0075】ソース色値S\_colorおよび行先色値D\_colorは、それらを表わすためにいくつのビットが使用されているかわからず、[0.0, 1.0]の範囲内の値をとる。各ビットエンコーディングはある範囲の値を表わす。たとえば、8ビット入力で14は、[14/256, 15/256]の範囲を表わす。計算を行なうときは、範囲全体を表わすための一つの値が選択される。もし範囲[14/256, 15/256]を表わすために値14/256が選択されたならば、計算のエラーはその範囲の下端に向けて偏らされるであろう。もし、その範囲の中間点が選択されたならば、すなわち14.5/256が選択されたならば、計算のエラーはその範囲に対して偏りがないようにされ、最終的な結果はより正確になるであろう。これは乗算器への入力の最下位ビットに1を連結することによって達成できる。したがって8ビットブレンドユニット178は、9ビット対9ビットの乗算器188および190を必要とする。

【0076】図10は、ブレンドオペレーション計算の間の中間値のフォーマットを示す。ディザ・オフセット値が、ピクセルのXアドレスおよびYアドレスの2つの最下位ビットに基づいて計算される。2つの積とディザオフセットとが加算される。最大で、積の値の一方の否定がとられてもよい。次に和が切捨てられ、クランプされて結果となる。

【0077】一実施例においては、図11およびここにその全体として引用により援用されるフォーリー（Foley）、ヴァンダム（vanDam）、フェイナー（Feiner）、およびヒュー（Hughes）による「コンピュータグラフィックスの原理および実際」（“Computer Graphics Principles and Practice”）、第2版p. 570～571に反映されるように、4×4 Bayerディザマトリックスのドット分散型組織的ディザアルゴリズムが用いられる。（-0.5, 0.5）の範囲のディザ値が切捨て前に結果値をオフセットする。もちろん、当分野で公知の他のディザアルゴリズムも使用できる。

【0078】図12は10ビットブレンドユニット180を示す。オペレーションごとに1つのピクセルをブレンドできるようにするためこれらのユニット4つが必要である。一実施例においては、10ビットブレンドユニット180各々と8ビットブレンドユニット178の各対との間で可能な限り多くの論理を再使用するような態様で、10ビットブレンドユニット180が実現される。10ビットブレンドユニット180は、8ビットブレンドユニット178に関して上に説明したのと同じ理由で2つの11ビット対11ビット乗算器188および190を必要とする。

【0079】この実施例はまた、そこを通過してソース係数Sfおよび行先係数DfのデータがROP/ブレンドユニット

166へ入力される2つの入力マルチプレクサ206および208を特徴とする。図14は、入力マルチプレクサ206および208の詳細な図である。

# 【0080】1.1.2 デプスユニット

図15はデプスユニット168のブロック図である。ピクセルALU120および121には2つのデプスユニット168がある。2つのデプスユニット168は、ソースデータ32ビットを行先データ32ビットと比較する。制御情報は16ビットマスクレジスタ210へ与えられ、次に、比較オペレーションの前にソースデータおよび行先データとビットごとに論理積を取られる。(1)フェール、(2)src<dest、(3)src==dest、(4)src<=dest、(5)src>dest、(6)src!=dest、(7)src==dest、および(8)バスの8つのテストのうち1つを指定する。ファンクション/オペレーションレジスタ212内の3ビットレジスタフィールドにより、符号なし整数比較オペレーションが指定される。正のIEEE単精度浮動小数点数であれば、符号ビットをゼロでマスクすれば、正しく比較されるであろう。

# 【0081】1.1.3 ステンシルユニット

図16および図17は、ステンシルユニット170のブロック図である。ピクセルALU120および121には2つのステンシルユニット170がある。2つの8ビットステンシルユニット170は各々、行先ステンシル用の8ビットデータ入力、1ビットデプステスト入力、8ビットデータ出力および1ビット比較出力を有する。符号のない整数比較オペレーションは、ファンクション/オペレーションレジスタ214内の3ビットレジスタフィールドにより指定され、(1)フェール、(2)ref<dest、(3)ref==dest、(4)ref<=dest、(5)ref>dest、(6)ref!=dest、(7)ref==dest、および(8)バスの8つのテストのうち1つを指定する。

【0082】図17を参照し、デプステストおよびステンシルテストの状態に依存して、3つのステンシルオペレーションコードのうち1つが選択される。オペレーションコードは、どのステンシルオペレーションが行なわれるかを決定する。可能なステンシルオペレーションは、dest、0、ref、wrap(dest+1)、wrap(dest-1)、saturate(dest+1)、saturate(dest-1)、およびdestである。

# 【0083】1.1.4 ウィンドウIDユニット

図18は、ウィンドウ識別（ID）ユニット172のブロック図である。ピクセルALU120および121には2つのウィンドウIDユニット172がある。2つの8ビットウィンドウID比較ユニット172は各々、行先WID用の8ビットデータ入力と1ビット比較結果出力とを有する。ウィンドウIDユニット172の挙動は、マスクレジスタ216内の8ビットフィールドと基準レジスタ218内の8ビットフィールドとファンクションレジスタ220内の3ビットフィールドとによって制御される。フ

アクションレジスタの2ビットフィールドは、(1)フェール、(2)ref<dest、(3)ref==dest、(4)ref<=dest、(5)ref>dest、(6)ref!=dest、(7)ref>=dest、および(8)バスの8つのテストのうち1つを指定する。ウィンドウIDユニット172とステンシルテストユニット170とは機能的に同一である。

【0084】ウィンドウIDユニット172、ステンシル

表2 ビットセルテスト

WIDテスト	アクションテスト	データテスト	アクション
フェール	--	--	データを書込まない
アドレス	フェール	--	アドレスのみを書込む
アドレス	アドレス	フェール	アドレスのみを書込む
アドレス	アドレス	アドレス	アドレス、データ、および色データを書込む

【0086】1.2 SRAMビットセルバッファ図19および図20は、SRAMビットセルバッファ118を示す。一実施例においては、SRAMビットセルバッファ118は、8ワード×1024ビットのマルチポートSRAMで実現される。グローバルバス122は、1024ビットパイプラインレジスタ127を通じてDRAMアレイ116のセンスアンプ160からSRAMビットセルバッファ118へデータを転送する1024ビット10ナノ秒読出バス125を含む。グローバルバス122はまた、1024ビットパイプラインレジスタ137を通じてSRAMビットセルバッファ118からセンスアンプ160へデータを転送する1024ビット10ナノ秒書込バス123を含む。この実施例においては、グローバルバス122はまた、マスク書込パイプラインレジスタ145およびマスク書込バス147を通じてSRAMビットセルバッファ118からセンスアンプ160へマスクデータ1024ビットを転送する。

【0087】SRAMビットセルバッファ118内のキャッシュラインは各々、バンクアドレス5ビットおよびコラムアドレス3ビットを含むタグ230と関連づけられる。タグ230は、現在SRAMビットセルバッファ118内にストアされているデータがやってきた位置を追跡するために使用される。

【0088】グローバルバスの読出オペレーションは、Read Cache Line (RL: キャッシュライン読出) オペレーションまたはChange Cache Line (CL: キャッシュライン変更) オペレーションのいずれかにより開始される。最初の10ナノ秒サイクルの間に、指定されたDRAMバンク158およびコラムからデータ読出パイプラインレジスタ127へとデータ1024ビットがコピーされる。次の10ナノ秒サイクルの間に、データ読出パイプラインレジスタ127からSRAMビットセルバッファ118

\* ユニット170およびデプスユニット168からのテスト結果は、デュアルビットセル3DRAMチップ110内のオペレーションを制御するため使用される。表2は、3つのユニットの結果に基づいて行なわれるかまたは行なわれないアクションのリストである。

【0085】

【表2】

内の指定されたラインへとデータ1024ビットがコピーされ、そのデータが取出されたバンクおよびコラムのアドレスがキャッシュラインのタグ230へ書込まれる。

20 【0089】グローバルバスの書込オペレーションは、Write Cache Line (WL: キャッシュライン書込) オペレーション、Masked Write Cache Line (ML: キャッシュラインのマスク書込) オペレーションまたはChange Cache Line (CL) オペレーションにより開始される。最初の10ナノ秒サイクルの間に、SRAMビットセルバッファ118内の指定されたラインからデータ書込パイプラインレジスタ137へとデータ1024ビットがコピーされ、ブレンマスクおよびバイトマスクレジスタからマスクデータ1024ビットが発生される。マスク書込パイプラインレジスタ147へコピーされる。もしオペレーションがWLまたはMLであれば、オペレーションのバンクおよびコラムアドレスは、バンクパイプラインレジスタ231およびコラムパイプラインレジスタ232へコピーされる。もしオペレーションがCLであれば、バンクパイプラインレジスタ231およびコラムパイプラインレジスタ232は、指定されたキャッシュラインのバンクおよびコラムタグ230からコピーする。次の10ナノ秒サイクルの間に、データ書込パイプラインレジスタ137の内容は、マスク書込パイプラインレジスタ145の制御下で、バンクおよびコラムパイプラインレジスタ231および232により指定されるDRAMバンクおよびコラムへコピーされる。

【0090】上に説明した手順に対するサンプル回数はいくつとありである。

【0091】

【数2】

```

33
function Mask[1023:0]
input [511:0] PlaneMask;
input [127:0] ByteMask;
input          IsMasked;
integer i;
begin
    for (i = 0; i < 1024; i = i+1) begin
        Mask[i] = (~IsMasked) & (PlaneMask[15512] &
        ByteMask[i/8]);
    end;
end;
end;
endfunction

```

【0092】ピクセルALU読出オペレーションは、Read Data (RDAT:データ読出) オペレーション、Read Pixel (RPIX:ピクセル読出) オペレーション、Single Pixel (SPIX:シングル・ピクセル) オペレーションおよびDual Pixel (DPIX:デュアル・ピクセル) オペレーションにより開始される。ピクセルALUデータバス141は、256ビット幅であり5ナノ秒の速度で動作し、SRAMピクセルバッファデータバス142もまた、256ビット幅であり5ナノ秒の速度で動作する。SRAMピクセルバッファ118は、512ビットのプレーンマスク222を含む付加的なラインを有する。MLオペレーションの間に、512ビットのプレーンマスクレジスタと128ビットのバイトマスクレジスタとが組合せられて、1024ビットのマスク書込145の内容が発生され、これは同時に書込データとしてラッチされる。

【0093】読出ポートおよび書込ポートは、128ビットのパウダリでアドレス指定される。256ビットデータチャネル138および142は200MHzで動作し、一方1024ビットグローバルバスチャネル122は100MHzで動作する。好ましい実施例においては、SRAMピクセルバッファ118は、8つのキャッシュライン224を有する。256ビットデータチャネル138および142は、任意の128ビットパウダリから始めて、キャッシュライン224の連続した256ビットにアクセスすることができなければならない。図3に示すように、一実施例においてこの要求は、SRAMピクセルバッファ118を、64ビットのデータ読出チャネル138a-dおよびデータ書込チャネル132a-dを備える4つの物理レイ118a-dに分けることによって満足される。この実施例においては、別個の読出アドレスおよび書込アドレスが、4つの別個のレイ各々に与えられる。他実施例においては、グローバルバス122が、SRAMピクセルバッファ118とセンスアンプ160との間での同時読出および書込オペレーションを可能にし、それによってデータタグの必要性をなくしている。

【0094】他実施例においては、データがDRAMレイ116から転送されてから、SRAMピクセルバッファキャッシュライン224のどのビットがピクセルALU120および121により更新されたかを示すため、SRAMピクセルバッファ内でデータタグ226が使用される。図20を参照し、データタグSRAM226は、16ワード×256ビットのデュアルポートSRAMで実現される。グローバルバス122は、256ビット10ナノ秒の読出/書込ポートに接続される。ピクセルALU120および121は、256個のビットごとの書込インテール (WE) を備えるマスクデータを256ビット5ナノ秒書込ポートに書込む。

【0095】Read Cache Line (RL) オペレーションの間に、データのラインは、SRAMピクセルバッファ118のデータ部に書込まれ、タグSRAM226内の対応するラインがクリアされる。

【0096】1.3 メモリ構成  
図21、図22および図23は、一実施例のDRAMレイ116の構成を示す。DRAMレイ116はモジュラードRAMバンク158を含む。メモリの基本単位は、1024ビットを保持するライン164である。ページ162は、8つのライン164、または8キロビットを含む。一実施例においては、DRAMバンク158は512個のページ162、またはメモリ4メガビットを含む。他実施例においては、DRAMバンク158は、1024個のページ162、またはメモリ8メガビットを含む。DRAMバンク158がメモリ8メガビットを保持する実施例においては、それぞれ40メガビット、48メガビット、64メガビット、80メガビット、96メガビット、128メガビットおよび160メガビットのDRAMレイ116をサポートするため、6、8、10、12、16および20個の個別のDRAMバンク158が必要である。

【0097】次の表は、64/72メガビットDirect RDRAM<sup>TM</sup>のデータシートおよび4メガビット×18SLDRAMデータシートの方両方に対するDRAMレイ116の構成に基づく。Direct RDRAM<sup>TM</sup>は、DRAMコアと1/0セクションとの間の128/144ビットインタフェースを有する。SLDRAMは、64/72ビットインタフェースを有する。デュアルピクセル3DRAM110は、1024ビットインタフェースを有する。

【0098】  
【表3】

表 3 DRAM 構成 (装置あたり)

	32	40	64	80	128	160
	1/2	1/2	1/2	1/2	1/2	1/2
DRAM あたりのバンク数	8	10	16	20	32	40
バンクあたりのページ数	512	512	512	512	512	512
ページあたりのワード数	8	8	8	8	8	8
ワードあたりのビット数	1024	1024	1024	1024	1024	1024

【0099】

\*10\*【表4】

表 4 DRAM バンクおよびページの可能な構成

ビット/装置	バンク/装置	ページ/装置	ビット/ページ	ワード/ページ	2レベル ワード
40 1/2ビット	10doubled	512	8K	44K	40K
	5	1024	8K	60K	40K
	10	512	8K	60K	80K
	5	512	16K	80K	80K
80 1/2ビット	20doubled	512	8K	84K	80K
	5	1024	16K	100K	80K
	10	1024	8K	120K	80K
	20	512	8K	160K	160K
	10	512	16K	160K	160K
	5	512	32K	160K	160K
160 1/2ビット	40doubled	512	8K	164K	160K
	5	1024	32K	180K	160K
	10	1024	16K	200K	160K
	20	1024	8K	240K	160K
	40	512	8K	320K	320K
	20	512	16K	320K	320K
	10	512	32K	320K	320K
	8	512	40K	320K	320K
5	512	64K	320K	320K	320K
	512	80K	320K	320K	320K
	4	512	80K	320K	320K

## 【0100】1.4 ビン構成

※DRAM<sup>TM</sup>と互換性のあるビン構成を示す。

デュアルビクスル3DRAMチップ110は、Direct RDRAM 【0101】

またはSLDRAMとビン互換性がある。次の表はDirect RDRAM<sup>TM</sup> 【表5】

表 5 デュアルビクスル 3DRAM ビン

信号	I/O	タイプ	記述
RQ[7:0]	I	RSL	制御およびアドレス情報
DQA[8:0]	I/O	RSL	データ入/出力A
DQB[8:0]	I/O	RSL	データ入/出力B
CEM	I	RSL	メモリからの知覚
CFMN	I	RSL	メモリからの知覚
CTM	I	RSL	メモリへの知覚
CTMN	I	RSL	メモリへの知覚
V <sub>REF</sub>			RSL 信号に対する論理しきい値基準電圧
V <sub>TERM</sub>			RSL 0-1のレベルに対するトリート電圧
SIO[1:0]	I/O	CMOS	シリアル入/出力
CHD	I	CMOS	シリアル入/出力
SCK	I	CMOS	シリアル入/出力
V <sub>DP</sub>			RDRAM 32およびバンクAに対する電源電圧
GND			RDRAM 32およびバンクAに対する接地基準

## 【0102】1.5 プロトコル

制御およびアドレス情報を送信するため、Direct RDRAM 50

は8つのピンを備え、SLDRAMは10個のピンを備える。デュアルビクスル3DRAMプロトコルは、いずれのイ

インタフェース技術においても使用できるよう8つのピンを必要とする。各ポートは、1つから4つのデュアルビタセル3DRAMチップ110を制御する。

【0103】次に図24および図25を参照し、制御およびアドレス情報800メガバイト/秒が、3つのチャネルに分割される。ビタセルALUオペレーションチャネル152は、1秒あたり400メガバイトを扱い、ビタセルALU120および121、SRAMビタセルバッファ118ならびにレンドリングバス112とのインタフェースを制御する。バンクオペレーションチャネル156は、1秒あたり200メガバイトを扱い、DRAMバンク158およびページ162へのアクセスおよびブリチャージを制御する。グローバルバスオペレーションチャネル154は、1秒あたり200メガバイトを扱い、グローバルバス122を通じてのSRAMビタセルバッファ118\*

表6 バンクオペレーション

0[10]	2ニツ	オペレーション	タイク
00	IDLE	7(イ)	1
01	PP	Precharge Page (ページリチャージ)	3
10	AP	Access Page (ページアクセス)	5
11	CP	Change Page (ページ変更)	5

【0106】図26を参照し、ビットD<sub>10</sub>が、共通バス114を通じて接続される4つの可能なデュアルビタセル3DRAM装置110のうち1つを選択する。ビットB<sub>10</sub>が、単一のデュアルビタセル3DRAM装置110内の32の可能なDRAMバンク158のうち1つを選択する。ビットP<sub>10</sub>が、DRAMバンク158内の1024個の可能なページ162のうち1つを選択する。

【0107】IDLEオペレーションは、1サイクルの間何もしない。図26および図27を参照し、Precharge Page (PP: ページリチャージ) オペレーションは、装置D<sub>10</sub>のバンクB<sub>10</sub>をブリチャージし、送信に3サイクル、実行に8サイクルかかる。

【0108】図26および図28を参照し、Access Page (AP: ページアクセス) オペレーションは、装置D<sub>10</sub>のバンクB<sub>10</sub>のページP<sub>10</sub>にアクセスし、送信に5サイクル、実行に8サイクルかかる。DRAMバンク158は、前もってブリチャージされていなければならない。

【0109】図26および図29を参照し、Change Page (CP: ページ変更) オペレーションは、Access Pageオペレーションと、その後続く同じ装置の同じバンクへのPrecharge Pageオペレーションとを組合せたものであ

\* キャッシュライン224の読出および書込を制御する。各チャネル152、154および156は、3つのチャネルが同時に異なるデュアルビタセル3DRAMチップ110に対して動作できるように、それ自身のオペレーションのフレーム化を行ないそれ自身の装置識別情報を含む。

【0104】1.5.1 DRAMバンクオペレーション  
DRAMバンクオペレーションは、帯域幅200メガバイト/秒の専用2ピン制御チャネル156を通じて送信される。DRAMバンクオペレーションは、任意のクロックの下がり端において送信を開始できる。次の表6に挙げるバンクオペレーションが、一実施例において規定される。

【0105】

【表6】

る。

【0110】1.5.2 グローバルバスオペレーション  
グローバルバスオペレーションの転送は、帯域幅が200メガバイト/秒の専用2ピン制御チャネル154により管理される。グローバルバスオペレーションは、任意のクロックの下がり端において送信を開始できる。グローバルバス転送には4クロック必要なので、コマンド送信にも4クロックかかり得る。

【0111】図30から図34を参照し、ビットD<sub>10</sub>は、共通バス114に繋がっている4つの可能なデュアルビタセル3DRAM装置110のうち1つを選択する。ビットB<sub>10</sub>は、装置110内の32の可能なDRAMバンク158のうち1つを選択する。ビットP<sub>10</sub>は、選択されたDRAMバンク158内の8つの可能なキャッシュラインバッファ160のうち1つを選択する。ビットL<sub>10</sub>は、SRAMビタセルバッファ118内の16の可能なキャッシュライン224のうち1つを選択する。

【0112】次の表7は、一実施例において規定されるグローバルバスオペレーションのリストである。

【0113】

【表7】

表7 グローバルバスオペレーション

0[2:0]	モード	オペレーション	サイクル
000	IDLE	アイドル	1
001	-	予約済	-
010	RL	Read Cache Line (キャッシュライン読込)	4
011	-	予約済	-
100	WL	Write Cache Line (キャッシュライン書込)	4
101	ML	Masked Write Cache Line (キャッシュラインのマスク書込)	4
110	FL	Flash Masked Write Cache Line (キャッシュラインのフラッシュマスク書込)	4
111	CL	Change Cache Line (キャッシュライン変更)	4

【0114】図30を参照し、IDLEオペレーションは、1サイクルの間何もしない。Read Cache Line (RL)、Write Cache Line (WL)、Masked Cache Line (ML) および Change Cache Line (CL) のコマンドはすべて、図31に示すように制御 (RQ) ピンにおいて同一のフォーマットを有する。図32に、Fast Fill CacheLine (FL: キャッシュラインの高速フィル) コマンドを示す。

【0115】図33を参照し、Read Cache Line (RL) オペレーションは、装置D<sub>0</sub>にのみ与えられ、DRAMバンク158、B<sub>0</sub>からパイプラインレジスタ127へキャッシュラインバッファ160、C<sub>0</sub>をコピーする。次に、SRAMビタセルバッファ118のラインL<sub>0</sub>へパイプラインレジスタ値を書込む。このオペレーションは、送信に4サイクル、各データ転送を行なうのに4サイクルかかる。SRAMビタセルバッファ118内の各ラインは、そのラインがどこから来たかを示すバンクおよびコラムタグ230を有する。バンクおよびコラムタグ230は、SRAMビタセルバッファ118への書込転送の間にRLオペレーションによってセットされる。

【0116】図34を参照し、Write Cache Line (WL) オペレーションは、装置D<sub>0</sub>にのみ与えられ、SRAMビタセルバッファ118からパイプラインレジスタ137へキャッシュライン224、L<sub>0</sub>をコピーする。パイプラインレジスタデータは、次にDRAMバンク158、B<sub>0</sub>のコラムC<sub>0</sub>へコピーされる。バンクおよびコラムタグ230は無視される。このオペレーションは、送信に4サイクル、各データ転送を行なうのに4サイクルかかる。

【0117】図35を参照し、Masked Write Cache Line (ML) オペレーションは、装置D<sub>0</sub>にのみ与えられ、SRAMビタセルバッファ118からパイプラインレジスタへキャッシュラインL<sub>0</sub>をコピーする。次に、パイプラインレジスタのデータは、バンクB<sub>0</sub>のコラムC<sub>0</sub>へコピーされる。バンクおよびコラムタグは無視される。このオペレーションは、送信に4サイクル、各データ転送を行なうのに4サイクルかかる。ブレンマスキレジスタ

タおよびバイトマスキレジスタは両方とも、1024ビットに拡張され、キャッシュラインがセンスアンプに書込まれるときビットごとの書込インエーブルとして使用される。

【0118】図36を参照し、Flash Masked Write Cache Line (FL: キャッシュラインのフラッシュマスク書込) オペレーションは、パイプラインデータおよびマスキが1つのコラムにだけでなく4つのコラムに書込まれるという点を除いてはMLと同様である。このオペレーションは、装置D<sub>0</sub>にのみ与えられ、SRAMからパイプラインレジスタへキャッシュラインL<sub>0</sub>をコピーする。次に、パイプラインレジスタのデータは、バンクB<sub>0</sub>のコラム0~3または4~7のいずれかへコピーされる。バンクおよびコラムタグは無視される。このオペレーションは、送信に4サイクル、各データ転送を行なうのに4サイクルかかる。ブレンマスキレジスタおよびバイトマスキレジスタは両方とも、1024ビットに拡張され、キャッシュラインがセンスアンプに書込まれるときビットごとの書込インエーブルとして使用される。

【0119】図37を参照し、Change Cache Line (CL) オペレーションは、装置D<sub>0</sub>にのみ与えられ、同時にRead Cache Line (RL) オペレーションおよびWrite Cache Line (WL) オペレーションを行なう。オペレーションからのバンクおよびコラムフィールドが、RLコマンドを制御する。キャッシュライン224からのバンクおよびコラムタグ230が、WLコマンドを制御する。このオペレーションは、送信に4サイクル、各データ転送を行なうのに4サイクルかかる。

【0120】1.5.3 ビタセルALUオペレーション  
ビタセルALUオペレーションは、帯域幅が400メガバイト/秒の専用4ピン制御チャネル152を通じて送信される。ビタセルALUオペレーションは、任意のクロックの立下り端で送信を開始できる。IDLE以外のオペレーションは、制御ピンを介する送信に2サイクル必要とする。各ビタセルALUオペレーションは、データピン (D



QA[<sub>0</sub>..]およびDQB[<sub>0</sub>..])を介する2サイクルでのデータ72ビットの転送または3サイクルでのデータ108ビットの転送を制御する。各オペレーションの2/3ビットはいくつかのサイクルが必要であることを示す。データ転送に3サイクル必要とするピクセルALUオペレーション \*

\*は、その後にIDLEサイクルが続かねばならない。次の表8に挙げるオペレーションは、一実施例について規定されるピクセルALUオペレーションである。

{0121}

{表8}

表8 ピクセルALUオペレーション

O {3:0}	二進法	オペレーション	ビット
0000	IDLE	アイドル	1
0001	-	予約済	
0010	-	予約済	2
0011	-	予約済	
0100	RREG	Read Register (レジスタ読出)	2
0101	-	予約済	
0110	WREG	Write Register (レジスタ書込)	2
0111	BREG	Broadcast Register (レジスタブロードキャスト)	2
1000	RDAT	Read Data (データ読出)	2
1001	-	予約済	
1010	WDAT	Write Data (データ書込)	2
1011	BDAT	Broadcast Data (データブロードキャスト)	2
1100	RPIX	Read Pixel (ピクセル読出)	2 または 3
1101	-	予約済	
1110	SPIX	Single Pixel (シングルピクセル)	2
1111	DPIX	Dual Pixel (デュアルピクセル)	2 または 3

{0122} ビットD<sub>0</sub>は、オペレーションのタイプを指定する。ビットD<sub>1</sub>は、共通バス114に繋がっている4つのデュアルピクセル3DRAM装置110のうち1つを選択する。ブロードキャストオペレーションにおいては、D<sub>1</sub>は無視される。ビットL<sub>0</sub>は、SRAMピクセルバッファ118内の8つのキャッシュライン224のうち1つを選択する。ビットP<sub>0</sub>は、キャッシュライン224内の16個のピクセルのうち1つを選択する。P<sub>0</sub>の解釈は、オペレーションのタイプおよび現在のピクセルのデプスに依って幅がある。ビットR<sub>0</sub>は、レジスタのアドレスを指定するためレジスタオペレーションにより使用される。

{0123} 図38を参照し、IDLEは、1サイクルの間何もしない。図39を参照し、Read Data (RDAT: データ読出) オペレーションは、指定された装置110のSRAMピクセルバッファ118の指定されたライン224の指定されたピクセルから生データ64ビットを読出す。このオペレーションは、いかなるレジスタのセッティングによっても影響されない。

{0124} Write Data (WDAT: データ書込) オペレーションは、指定された装置110のSRAMピクセルバッファ118の指定されたライン224の指定されたピクセルへ生データ64ビットを書込む。バイトごとの書込イ

ネーブルが、ピンDQA<sub>0</sub>およびDQB<sub>0</sub>を通じて送信される。このオペレーションはいかなるレジスタのセッティングによっても影響されない。

{0125} Broadcast Data (BDAT: データブロードキャスト) オペレーションは、すべてのデュアルピクセル3DRAM装置110のSRAMピクセルバッファ118の指定されたライン224の指定されたピクセルへ生データ64ビットをブロードキャストする。バイトごとの書込イネーブルが、ピンDQA<sub>0</sub>およびDQB<sub>0</sub>を通じて送信される。このオペレーションはいかなるレジスタのセッティングによっても影響されない。

{0126} 図40を参照し、Read Register (RREG: レジスタ読出) オペレーションは、指定された装置110からレジスタ値を讀出す。

{0127} Write Register (WREG: レジスタ書込) オペレーションは、レジスタ値を指定された装置110へ書込む。バイトごとの書込イネーブルが、ピンDQA<sub>0</sub>およびDQB<sub>0</sub>を通じて送信される。

{0128} Broadcast Register (BREG: レジスタブロードキャスト) オペレーションは、レジスタ値をすべての装置110へブロードキャストする。バイトごとの書込イネーブルが、ピンDQA<sub>0</sub>およびDQB<sub>0</sub>を通じて送信される。

【0129】図41を参照し、Read Pixel (RPIX) オペレーションは、表示リフレッシュのためにバックされたピクセルを讀出す。

【0130】Single Pixel (SPIX) オペレーションは、単一のソースピクセルを既にSRAMピクセルバッファ118内にあるピクセルデータとマージする。

【0131】Dual Pixel (DPIX) オペレーションは、2つの隣り合ったソースピクセルを既にSRAMピクセルバッファ118内にあるピクセルデータとマージする。

【0132】デュアルピクセル3DRAMチップ110の一実施例は、ダーティタグを使用する。この実施例においては、Write Tag (WTAG: タグ書込) オペレーションは、バイトマスクデータ64ビットを128ビットのマスクに拡張し、これは指定された装置のダーティタグ226の指定されたライン224の上位半分または下位半分に書込まれる。ラインに対するダーティタグ226は、グローバルバスを通じてDRAMアレイにラインを書込むのにかかる時間の量と一致して、2サイクルで書込むことができる。

【0133】図42は、デュアルピクセル3DRAM装置110が各サイクルにおいてデータ36ビットを転送できる一実施例を示す。データの2クロックまたは3クロックが、各ピクセルALUオペレーションに関連付けられる。

【0134】図43を参照し、Read Data (RDAT) オペレーションおよびRead Pixel (RPIX) オペレーションは、SRAMピクセルバッファ118ならびにピクセルALU120および121のいくつかを用いる。ピクセルALU120および121が使用されず、SRAMピクセルバッファ118には何も書込まれない。図43は、3サイクルRPIXが後に続く2サイクルRDATを示す。

【0135】図44に示すように、Write Data (WDAT) オペレーション、Broadcast Data (BDAT) オペレーション、Write Register (WREG) オペレーションおよびBroadcast Register (RREG) オペレーションは、DQピン上でデータを受信し、ピクセルALU120および121を通じてデータを送り、適当な段でデータをラップする。上述のオペレーションのいずれも3サイクル転送による利益を受けない。

【0136】Single Pixel (SPIX) オペレーションおよびDual Pixel (DPIX) オペレーションは、SRAMピクセル

バッファ118ならびにピクセルALU120および121を十分に利用する。これらの2つのオペレーションは、SRAMピクセルバッファ118からデータを読出してアンパックし、DQピンからデータを受信し、ピクセルALU120および121内のデータの組を両方組合せ、再びSRAMピクセルバッファ118へ結果を書込む。図45は2サイクルでのSPIX転送を示し、図46は3サイクルでのDPIX転送を示す。

【0137】1.6 オペレーションタイミング上に説明したオペレーションの多くのタイミングの例を図47から図53に示す。

【0138】図47は、複合2サイクル読出および2サイクル書込オペレーションの図である。

【0139】図48は、複合2サイクル読出および3サイクル書込オペレーションの図である。

【0140】図49は、複合3サイクル読出および2サイクル書込オペレーションの図である。

【0141】図50は、複合3サイクル読出および3サイクル書込オペレーションの図である。

【0142】図51は、4つの2サイクル読出オペレーションを行なうために必要とされるすべてのオペレーションの図である。

【0143】図52および図53は、8つの2サイクルDual Pixel (DPIX) オペレーションを行なうために必要とされるすべてのオペレーションを示す。DPIXオペレーション4〜7は、DPIXオペレーション0〜3に対するバンクとは異なるバンク158へのものである。

【0144】1.7 レジスタ

RREGオペレーション、WREGオペレーションおよびBREGオペレーションは、128ワード×64ビットレジスタアドレスベースに対応したものである。ピクセルALU120および121を再プログラムするため必要とされるサイクルの数を最小限にするため、レジスタの幅を利用して効率化が図られる。次の表9は、デュアルピクセル3DRAMチップ110において用いられるレジスタを識別するものである。図54は、表9に挙げられるいくつかのレジスタのデータフォーマットを示す。影をつけた部分はこれらのレジスタ用に予約されたフィールドを表わす。

【0145】

表9]

表 9 レジスタマップ

R <sub>レジ</sub>	名前	ID	初期値	ビット
0	Identification	ID	N/A	R
1	Feature Enable	FE	0x0000 0000 0000 0000	R/W
2	Pixel Config	PC	0x0000 0000 0000 0000	R/W
3	Stencil Depth Config	SDC	0x0000 0000 0000 0000	R/W
5-4	ColorOp[1:0]	CO	0x0000 0000 0000 0000	R/W
6	Constant Color	CC	0x0000 0000 0000 0000	R/W
13-7	予約済	-	-	-
15-14	Byte Mask [1:0]	BM	0xFFFF FFFF FFFF FFFF	R/W
23-16	Plane Mask [7:0]	PM	0xFFFF FFFF FFFF FFFF	R/W
63-23	予約済	-	-	-
67-64	ColorWIDLAT [3:0]	CWL	0x0000 0000 0000 0000	R/W
71-68	OverlayWIDLAT [3:0]	OWL	0x0000 0000 0000 0000	R/W
72	Display Config	DC	0x0000 0000 0000 0000	R/W
127-73	予約済	-	-	-

## 【0146】1.7.1 識別

この読出専用レジスタは、チップのマスキング、バージョン、部品番号および製造者を識別する。

## 【0147】1.7.2 FeatureEnable

このレジスタは、デュアルピクセル3DRAMの将来のバージョンにおける新しい機能を可能または不能にする。デュアルピクセル3DRAMの初期のバージョンにおいては、これは、0x0000#0000#0000#0000にリセットされ、他のいかなる値にもセットされるべきではない。

## \* 【0148】1.7.3 PixelConfig

図55は、PixelConfigレジスタのデータフィールドフォーマットを示す。このレジスタは、ピクセルのデプスおよびピクセルの詳細なフォーマットを決定する。影つきで図示される予約されたフィールドは、将来の互換性を確保するために、0にセットされなければならない。表10は、レジスタのデータフィールドを説明する。

## 【0149】

【表10】

表 10 PixelConfig レジスタフィールド

フィールド	幅	記述
PixelSize	3ビット	SRAM→PALU および PALU→SRAMのビット数
ColorMode	4ビット	色データが付けられる態様を決定する
BufferSelect	2ビット	A/B 色データを選択する
InputMode	4ビット	DQ→PALU レーティングを制御する
DestinationFactor	4ビット	行先ビット係数
SourceFactor	4ビット	ソースビット係数
WIDMask	8ビット	WID マスク値
WIDRef	8ビット	WID 基準値
WIDFunc	3ビット	WID 比較レーン

【0150】PixelSizeフィールドは、SPIXオペレーションおよびDPXオペレーションの間にピクセルサイズを選択するため、ピクセルALUからSRAMへのフォーマット140およびSRAMからピクセルALUへのフォーマット144により使用される。

## 【0151】

【表11】

表 11 ピクセルサイズ

PS[2:0]	ビット数
0	8ビット, 16ビット, 32ビット
1	64ビット
2	128ビット
3	256ビット
4	512ビット

【0152】ColorModeフィールドは、フォーマット140および144用の色データフォーマットを指定す

る。表12は色モードフォーマットを説明する。

## 【0153】

【表12】

表 12 ColorMode フィールド

レーン	色データ	赤	緑	青
7:0	7:0	7:0	7:0	7:0
0	8	8	8	8
1	8	0	0	0
2	2	10	10	10
3	10	10	10	10
4	4	4	4	4
5	8	8	0	0
6	0	5	6	5
7	1	5	5	5

【0154】BufferSelectフィールドは、ピクセルALU120および121とSRAMピクセルバッファ118との間でデータをフォーマット化するフォーマット140お

よび144のための色バッファを選択する。BufferSelect[0]は、32ビットワード内の色の下位16ビットと上位16ビットとのいずれかを選択する。もし、色が32ビットまたは40ビットであれば、BufferSelect[0]は無効である。BufferSelect[1]は、ピクセル内に色64ビットまたは80ビットがあるとき、色データの下部もしくは上位32ビットまたは下部もしくは上位40ビットを選択する。BufferSelect[1]は、もしPixelSizeが\*

\*64ビットであれば無効である。

【0155】InputModelは、入力データフォーマット130を制御する。ソース係数フィールドおよび行き先係数フィールドは、表13に説明するようにエンコードされる。

【0156】

【表13】

表13 ブレンドオペレーションフィールド

エンコード	係数				OpenGL
	7A7f	赤	緑	青	
0	0				GL_ZERO
1	1				GL_ONE
2	1 min (Sa, 1-Da)				GL_SRC_ALPHA_SATURATE
4	Sa	SR	Sg	Sb	GL_SRC_COLOR
5	1-Sa	1-SR	1-Sg	1-Sb	GL_ONE_MINUS_SRC_COLOR
6	Sa				GL_SRC_ALPHA
7	1-Sa				GL_ONE_MINUS_SRC_ALPHA
8	Da	DR	Dg	Db	GL_DST_COLOR
9	1-Da	1-DR	1-Dg	1-Db	GL_ONE_MINUS_DST_COLOR
10	Da				GL_DST_ALPHA
11	1-Da				GL_ONE_MINUS_DST_ALPHA
12	Ca	CR	Cg	Cb	GL_CONSTANT_COLOR
13	1-Ca	1-CR	1-Cg	1-Cb	GL_ONE_MINUS_CONSTANT_COLOR
14	Ca				GL_CONSTANT_ALPHA
15	1-Ca				GL_ONE_MINUS_CONSTANT_ALPHA

【0157】WIDFuncフィールド、DepthFuncフィールドおよびStencilFuncフィールドは、表14に説明するようにエンコードされる。

※【0158】

【表14】

※  
表14 WIDFunc, DepthFunc, および StencilFunc 7f-ff

DepthFunc	条件	OpenGL
0	Pass	GL_ALWAYS
1	Source > Destination	GL_GREATER
2	Source == Destination	GL_EQUAL
3	Source >= Destination	GL_GEQUAL
4	Fail	GL_NEVER
5	Source <= Destination	GL_LEQUAL
6	Source != Destination	GL_NOTEQUAL
7	Source < Destination	GL_LESS

【0159】1.7.4 StencilDepthConfig

図56は、StencilDepthConfigレジスタのデータフィールドフォーマットを示す。このレジスタは、ステンシルユニット170およびデプスユニット168を制御する。16ビットのデプスマスクフィールド、3ビットのデプス比較ファンクションフィールド、および1ビットのDepthLoadフィールドが、デプスユニット170を制御する。StencilMaskフィールドが、ステンシル比較オペレーションにおいてどのビットが関係するかを決定す

る。行き先データがStencilRef値と比較される。StencilFuncが、どのように行き先値と基準値とが比較されるかを指定する。「StencilOp」で始まるフィールドが、新しいステンシルデータをどのように計算するかを決定する。影つきで図示される予約されたフィールドは、将来の互換性を確保するために、0にセットされなければならない。

【0160】

【表15】

表 15 StencilDepthConfig レジスタ

フィールド	幅	記述
DepthMask	16 ビット	デプスのどのビットが比較されるかを制御する
DepthFunc	3 ビット	比較オペレーションを選択する
DepthLoad	1 ビット	DEPTH の値に入力レジスタにロードする
StencilMask	8 ビット	ステンシルマスク
StencilRef	8 ビット	ステンシル基準値
StencilFunc	3 ビット	ステンシル比較オペレーション
StencilOpFail	3 ビット	もしステンシルがフェールであればステンシルオペレーション
StencilOpZFail	3 ビット	もしステンシルがフェールでデプスがフェールであればステンシルオペレーション
StencilOpZPass	3 ビット	ステンシルがフェールでデプスがフェールであればステンシルオペレーション

【0161】DepthFuncフィールドおよびStencilFuncフィールドは、前掲の表14によりエンコードされる。 【0163】

【0162】「StencilOp」で始まるフィールドは、次 \* 【表16】  
表 16 StencilOp で始まるフィールド

StencilOp	オペレーション	OpenGL
0	Destination	GL_KEEPP
1	0	GL_ZERO
2	Reference	GL_REPLACE
3	Destination	GL_INVERT
4	Saturate(Destination+1)	GL_INCR
5	Saturate(Destination-1)	GL_DECR
6	Destination+1	GL_INCRWRAP
7	Destination-1	GL_DECRWRAP

【0164】1.7.5 ColorOP[0]

図57は、ColorOP[0]レジスタフィールドを示す。ColorOP[0]レジスタは、アルファ、赤、緑および青ROPおよびブレンドユニット174および176の個々のための制御フィールドを有する。影つきで図示される予約されたフィールドは、将来の互換性を確保するため、0にセットされなければならない。

【0165】

【表17】

表 17 ColorOp[0] レジスタ

フィールド	幅	記述
AlphaLogicOp	8 ビット	787α論理オペレーション
AlphaBlendOp	3 ビット	787αブレンドオペレーション
RedLogicOp	8 ビット	赤論理オペレーション
RedBlendOp	3 ビット	赤ブレンドオペレーション
GreenLogicOp	8 ビット	緑論理オペレーション
GreenBlendOp	3 ビット	緑ブレンドオペレーション
BlueLogicOp	8 ビット	青論理オペレーション
BlueBlendOp	3 ビット	青ブレンドオペレーション

【0166】論理オペレーションフィールドは、ソース色、行き先色およびパターン色の256個のビットごとのブール演算のうち1つを選択するため、Microsoft Windows<sup>TM</sup> 規約によってエンコードされる。パターンデータは、ConstantColorレジスタから来る。

【0167】ブレンドオペレーションフィールドは、表18で説明するようにエンコードされる。

【0168】

【表18】

表 10 ブレンドオペレーションフィールド

BlendOp	オペレーション	OpenGL
0	LogicOp (Sc, Dc, Pattern)	GL_COLOR_LOGIC_OP
1	Sc+Sc+Dc+Dc	GL_FUNC_ADD
2	Sc+Sc-Dc+Dc	GL_FUNC_SUBTRACT
3	Dc+Dc-Sc+Sc	GL_FUNC_REVERSE_SUBTRACT
4	min(Sc, Dc)	GL_MIN
5	max(Sc, Dc)	GL_MAX

## 【0169】 1.7.6 ColorOP[1]

図5 8は、ColorOP[1]レジスタのデータフィールドフォーマットを示す。ColorOP[1]レジスタは、アルファ、赤、緑および青ROPおよびブレンドユニット176個々のための制御フィールドを有する。影つきで図示される\*

\* 予約されたフィールドは、将来の互換性を確保するため0にセットされねばならない。表19は、ColorOP[1]レジスタのフィールドを説明する。

## 【0170】

## 【表19】

表 19 ColorOp[1] レジスタ

フィールド	幅	記 述
AlphaSize	4 ビット	アルファのサイズ
AlphaLoad	1 ビット	DPX の間にアルファ値にロードする
AlphaBlendEnable	1 ビット	ブレンド用にアルファ入力をフォーマット化する
AlphaDitherEnable	1 ビット	アルファのデザイザリングを可能化する
RedSize	4 ビット	赤のサイズ
RedLoad	1 ビット	DPX の間に赤値にロードする
RedBlendEnable	1 ビット	ブレンド用に赤入力をフォーマット化する
RedDitherEnable	1 ビット	赤のデザイザリングを可能化する
GreenSize	4 ビット	緑のサイズ
GreenLoad	1 ビット	DPX の間に緑値にロードする
GreenBlendEnable	1 ビット	ブレンド用に緑入力をフォーマット化する
GreenDitherEnable	1 ビット	緑のデザイザリングを可能化する
BlueSize	4 ビット	青のサイズ
BlueLoad	1 ビット	DPX の間に青値にロードする
BlueBlendEnable	1 ビット	ブレンド用に青入力をフォーマット化する
BlueDitherEnable	1 ビット	青のデザイザリングを可能化する

【0171】成分サイズフィールドは表20に説明するようにエンコードされる。

## 【0172】

## 【表20】

表 20 サイズフィールドのエンコーディング

エンコーディング	サイズ
1	1 ビット
2	2 ビット
3	3 ビット
4	4 ビット
5	5 ビット
6	6 ビット
7	7 ビット
8	8 ビット
9	9 ビット
10	10 ビット

## 【0173】 1.7.7 ConstantColor

図5 9は、ConstantColorレジスタのデータフィールドフォーマットを示す。ConstantColorレジスタは、論理オペレーション用のパターンデータまたはブレンドオペレーション用の定数データのいずれかとして使用される。影つきで図示される予約されたフィールドは、将来の互換性を確保するためセットされねばならない。表2

1は、ConstantColorレジスタのフィールドを説明する。

## 【0174】

## 【表21】

表 21 Constant Color レジスタ

フィールド	幅	記 述
AlphaConstant	10 ビット	アルファ定数
RedConstant	10 ビット	赤定数
GreenConstant	10 ビット	緑定数
BlueConstant	10 ビット	青定数

## 【0175】 1.7.8 Byte Mask[1:0]

これら2つのレジスタは、128ビットのバイトマスクへの読出/書込アクセスを可能にする。バイトマスクは、MLオペレーションおよびFLオペレーションに影響する。

## 【0176】 1.7.9 Plane Mask[7:0]

これら8つのレジスタは、512ビットのプレーンマスクへの読出/書込アクセスを可能にする。プレーンマスクは、MLオペレーション、FLオペレーション、SPIXオペレーションおよびDPXオペレーションに影響する。

## 【0177】 1.7.10 ColorWIDOUT[3:0]

これら4つのレジスタは、Aパッパ(0)またはBパッパ

ア(1)のαRGB色データを選択するためウィンドウIDルックアップテーブルにロードする。表示リフレッシュの間、8ビットのWIDTHは、選択ビットを生成するための256エントリルックアップテーブル(LUT)へのインデックスとなる。

#### 【0178】1.7.11 OverlayWIDLU[3:0]

これら4つのレジスタは、Aバッファ(0)またはBバッファ(1)のオーバーレイデータを選択するためウィンドウIDルックアップテーブルにロードする。表示リフレッシュの間、8ビットWIDTHは、選択ビットを生成するための256エントリLUTへのインデックスとなる。

#### 【0179】1.7.12 DisplayConfig

図60は、DisplayConfigレジスタのデータフィールドフォーマットを示す。このレジスタは、RPIX命令の間にピクセルデータの処理を制御するため、3ビットのピクセルサイズフィールドを有する。影つきで図示される予約されたフィールドは、将来の互換性を確保するため0にセットされなければならない。PixelSizeフィールドは前掲の表1によりエンコードされる。

#### 【0180】1.8 高速領域クリア

MLオペレーション、FLオペレーション、APオペレーション、PPオペレーション、WREGオペレーションおよびBREGオペレーションは、領域を極めて迅速にクリアするため、あわせて使用することができる。ByteMask[1:0]レジスタは、キャッシュライン内の個々のピクセルを書込用に可能化または不能化するため使用できる。PlaneMask[7:0]レジスタは、書込用にピクセル成分を可能化または不能化するため使用できる。

【0181】もし、ByteMask[1:0]レジスタを使用して個々のピクセルをマスクする必要があるのであれば、APオペレーション、PPオペレーション、WREGオペレーションおよびMLオペレーションを使用しなければならない。いずれのピクセルもまだフィルされないうちに、1024ビットキャッシュライン中のすべてのピクセルが所望のクリア値にセットされ、512ビットPlaneMaskレジスタが、クリアされるべきピクセルのこれらのビットへの書込のみを可能化するようにセットされる。次に、WREGオペレーションまたはBREGオペレーション2つを使用して、128ビットByteMaskレジスタが書込される。次に、MLオペレーションが、ByteMaskレジスタおよびPlaneMaskレジスタを使用して、キャッシュラインをセンスアンプに書込む。続くWREGおよびMLオペレーションシーケンスは、10ナノ秒ごとに128バイトまたは12.8ギガバイト/秒のピークフィル速度をサポートするように重ね合わせる。80メガビットの装置は、819マイクロ秒でフィルされ得る。

【0182】もしページ内のすべてのピクセルをクリアするのであれば、128ビットByteMaskレジスタはオール1にセットでき、WREG、MLオペレーションシーケンスの代わりにFLオペレーションシーケンスを使用できる。

1024ビットキャッシュラインおよび512ビットPlaneMaskレジスタは前と同様セットされる。FLオペレーションごとにページの半分がクリアされ、したがって、ページ全体は20ナノ秒でクリアされ得る。ピークフィル速度は、20ナノ秒ごとに8×128バイトまたは51.2ギガバイト/秒である。80メガビットの装置は、205マイクロ秒でフィルされ得る。表22は、すべてのピクセルサイズについてのピークフィル速度を示す。図61および図62は、上に説明した高速フィル(Fast Fill)および非常に高速のフィル(Really Fast Fill)のオペレーションシーケンスを示す。

#### 【0183】

#### 【表22】

表22 高速領域クリアピーク速度

ピクセルサイズ	MLフィル速度	FLフィル速度
8ビット	12.8 GP/s	51.2 GP/s
16ビット	6.4 GP/s	25.6 GP/s
32ビット	3.2 GP/s	12.8 GP/s
64ビット	1.6 GP/s	6.4 GP/s
128ビット	800 MP/s	3.2 GP/s
256ビット	400 MP/s	1.6 GP/s
512ビット	200 MP/s	800 MP/s

#### 【0184】2.0 データルーティング

デュアルピクセル3DRAMチップ110の多数の新規な特徴、および、そのチップを基礎とするグラフィクスシステムが、チップ110の4つのデータフォーマット130、134、140および144によって実装される。本開示のこのセクションを通じて、SRAMピクセルバッファ118を2つの個別の機能ブロックとして説明する。なぜなら、ここに開示する処理モードおよびルーティングモードの多くが2つのピクセルを同時に処理する動作モードに関連しているためである。したがって、SRAMピクセルバッファ118は、機能的に2つの部分に分割して考えることができるものと理解されたい。

#### 【0185】2.1 入力データフォーマット

図63を参照して、このセクションでは、I/Oバス124を介してピクセルALU120および121へと入来するデータのルーティングおよび処理について説明する。ピクセルALU書込動作は、2クロック内でデータの72ビットを、または、3クロック内でデータの108ビットを、処理されるピクセルのフォーマットおよびサイズに応じて転送する。入来データはできるだけピンの近くの入力データデマルチプレクサ126を通過して、72ビットまたは108ビットのいずれかとして並行にピクセルALUに与えられる。ピクセルALU120および121は、ソースデータとして以下の入力を行う: Alpha[10:0], Red0[10:0], Green0[10:0], Blue0[10:0], Depth0[31:0], Alpha1[10:0], Red1[10:0], Green1[10:0], Blue1[10:0], および Depth1[31:0]。ルーティングは、4ビットレジスタフィールドによって制御される。

#### 【0186】2.1.1 ピクセル圧縮

三次元ピクセルマージを行なう場合、レンダリングコントローラは、色およびデプス情報のみを送信すればよい。というのは、ステンシルおよびウィンドウID情報はレジスタ内にストアすることができるためである。三角形または表面パッチをレンダリングする場合、生成されたピクセルは通常、高レベルのコヒーレンシーを示す。このセクションでは、最小数のクロックサイクル内でデータピンを通じてピクセルペアを送信する、新規な無損失の圧縮方式について説明する。

【0187】この方式は、ピクセルペア間の差および、最も最近処理された(古い)ピクセルペアと入ってくる(新しい)ピクセルペアとの間の差を評価する。1対のピクセル間には、また、2対の連続して処理されるピクセルペア間には、高レベルのコヒーレンシーが存在することがしばしばあるため、入来るピクセルデータは時として、非常に少ない数のビットで表わすことができる場合がある。このような状況下では、送信すべきなのは古いピクセルペアと新しいピクセルペアとの差のみであって、この差は、新しいピクセルペアの最下位ビットで表わされる。最良の場合、すなわち、ピクセルペア間に高レベルのコヒーレンシーが見られる場合、2:1の圧縮比を達成することが可能であり、レンダリングコントローラ102からデュアルピクセル3DRAMチップ110への入力帯域幅を有効に2倍に増すことができる。中程度のコヒーレンシーが見られる場合には、4:3の圧縮比を達成することができる。コヒーレンシーがほとんど見られない場合、圧縮を行なうことはできない。

【0188】デュアルピクセル3DRAM110は、最も新しく送られてきたピクセルペアを、各デプス値につき32ビットおよび4つの色成分の各々につき10ビットで、レジスタの組内にストアする。したがって、1ピクセルあたりデータの72ビットがレンダリングパス112を介して送信される。ピクセルはデュアルピクセル3DRAMチップ110ではしばしば2対で処理されるため、1対のピクセルのための完全なデータは144ビットで表わされる。テクスチャマッピングは色値のコヒーレンシー\*

$$\begin{aligned} DAO &= NAO - OAO; DA1 = (NA1 - OA1) - (NAO - OAO); \\ DRO &= NRO - ORO; DR1 = (NR1 - OR1) - (NRO - ORO); \\ DGO &= NGO - OGO; DG1 = (NG1 - OG1) - (NGO - OGO); \\ DBO &= NBO - OBO; DB1 = (NB1 - OB1) - (NBO - OBO); \\ DZO &= NZO - OZO; DZ1 = (NZ1 - OZ1) - (NZO - OZO); \end{aligned}$$

【0195】算出された差のみが、レンダリングコントローラ102からデュアルピクセル3DRAMチップ110へと送信されるので、レンダリングパス112を介して送信されるピクセルあたりのビット数が低減される。新しいピクセル成分は、デュアルピクセル3DRAMチップ110上で入力データフォーマット130により、以下のように再生される。

【0196】

\*一を低下させる傾向にあるため、色値を圧縮することによって帯域幅をセーブすることはできないであろう。しかし、デプス値はほとんどの場合、高レベルのコヒーレンシーを示す。したがって、ここに開示する圧縮方式は、デプス値間の高レベルのコヒーレンシーを利用する。以下に、ピクセルデータを圧縮および伸長するためのアルゴリズムを説明する。

【0189】レンダリングコントローラ102およびデュアルピクセル3DRAMチップ110は両方とも、最も新しく送られてきたピクセルペアをストアしている。以下に列記するのは、「古い」ピクセル0および1を色(アルファ、赤、緑および青)成分およびデプス成分に分解したものである。

【0190】

【数3】

$$\begin{aligned} OAO, ORO, OGO, OBO, OZO \\ OA1, OR1, OG1, OB1, OZ1 \end{aligned}$$

【0191】レンダリングコントローラ102はその通常のレンダリング処理の一部として、新しいピクセルペアを算出する。多くの場合、古いピクセル成分と新しいピクセル成分とは同様の値を有し、新しいピクセルペアもまた同様の値を有する。下に、「新しい」ピクセル0および1を色(アルファ、赤、緑および青)成分およびデプス成分に分解したものを示す。

【0192】

【数4】

$$\begin{aligned} NAO, NRO, NGO, NBO, NZO \\ NA1, NR1, NG1, NB1, NZ1 \end{aligned}$$

【0193】もしピクセル成分が同様の値を有する場合、それらの差は小さく、成分自体よりも少ないビットで表わすことができる。ピクセル成分の差は、レンダリングコントローラ102によって以下の等式を使用して計算される。式中、頭に付された「D」は、差(difference)またはデルタ(delta)を表わす。

【0194】

【数5】

$$\begin{aligned} NAO &= OAO + DAO; NR1 = OA1 + DA0 + DA1 \\ NRO &= ORO + DRO; NR1 = OR1 + DRO + DR1 \\ NGO &= OGO + DGO; NG1 = OG1 + DGO + DG1 \\ NBO &= OBO + DBO; NB1 = OB1 + DBO + DB1 \\ NZO &= OZO + DZO; NZ1 = OZ1 + DZO + DZ1 \end{aligned}$$

【数6】

50 【0197】図64を参照して、入力データフォーマット



タ130は、その伸長方式を3層で実現する。第1の層では、種々のフォーマットから10個の差成分を抽出して、必要であればそれらを符号拡張する。第2の層では、それらの差を先のピクセル成分に加えて、新しいピクセル成分を再生する。第3の層では、新しいピクセル成分をフォーマット化する。色成分は、その最終幅にしたがって左にシフトされ、デプス値の上位16ビットがマスクされる。

#### 【0198】2.1.2 入力データフォーマット

下の表23に記載するように、デュアルピクセル3DRAM 1110は、以下のような入力データフォーマットを有する。もしオペレーションがWDATまたはBDATである場合、入力フォーマットは、レジスタのプログラム方法にかかわらず、強制的にモード0にされる。オペレーションが\*

\*SPIXである場合、入力フォーマットは強制的にモード7にされる。オペレーションがDPIXの場合、入力フォーマットは、InputModeレジスタフィールドに書き込むことによって設定される。モード0および1は、実行が不要な二次元のピクセル更新の場合に使用される。モード2は、アンチエイリアシングオペレーション中に使用される。モード4および5は、三次元のピクセル更新に使用される。図65は、レンダリングコントローラ102から入力データフォーマット130へとレンダリングパス112を介して送信されるデータについて、可能性のあるすべてのフォーマットを示す。

#### 【0199】

【表23】

表23 入力データフォーマット内のビットフィールド割当て

モード	オペレーション	色成分	16ビット 速度	7577				赤				緑				青				デプス			
				DR0	DA1	DR0	DA1	DG0	DA1	DG0	DA1	DB0	DA1	DB0	DA1	DZ0	DA1	DZ0	DA1	DZ0	DA1	DZ0	DA1
0	WDAT	2	400	8	8	8	8	8	8	8	8	8	8	8	8	-	-	-	-	-	-	-	-
	BDAT	3	267	10	10	10	10	10	10	10	10	10	10	10	10	-	-	-	-	-	-	-	-
	DPIX	2	400	-	-	-	-	-	-	-	-	-	-	-	-	32	32	-	-	-	-	-	-
4	DPIX	2	400	6	4	6	4	6	4	6	4	6	4	6	4	18	14	-	-	-	-	-	-
	DPIX	3	267	8	6	8	6	8	6	8	6	8	6	8	6	24	24	-	-	-	-	-	-
	DPIX	2	400	2	2	7	5	7	5	7	5	7	5	7	5	18	14	-	-	-	-	-	-
5	DPIX	3	267	2	2	10	7	10	7	10	7	10	7	10	7	24	24	-	-	-	-	-	-
	SPIX	2	200	10	-	10	-	10	-	10	-	10	-	10	-	32	-	-	-	-	-	-	-

#### 【0200】2.1.3 アクムレータ

図66は、入力データフォーマットのアクムレータ層の実現を図示する。同じ10ビット色アクムレータ設計が、アルファ、赤、緑および青データを処理するのに使用される一方、同様に設計された32ビットアクムレータが、デプスデータを処理するのに使用される。WDAT、BDAT、およびSPIXオペレーション中には、差分データがアクムレータ内に直接ロードされる。DPIXオペレーション中には、差分データのローディングはレジスタビットによって制御される。

#### 【0201】2.1.4 最終フォーマット

図67は、入力データフォーマット130の最終フォーマット層を図示する。色成分は、ROP/Blendユニット166によって使用されるであろうビット数にしたがって左にシフトされる。もし成分がブレンドされる場合、1が最下位ビットの右にアペンドされる。以下に示すのは、入力データフォーマット130内で処理の最終層のverilog関数である。

#### 【0202】

【数7】

```

59
function [10:0] FormatColor;
input [9:0] Data; // Raw color component data
input [3:0] Size; // Size of color component
input
Blend; // Set if colors are to be blended
begin
  case (Size)
    4'b0001: FormatColor = {Data[0], Blend, 9'b000000000}; // 1 bit
    4'b0010: FormatColor = {Data[1:0], Blend, 8'b000000000}; // 2 bits
    4'b0011: FormatColor = {Data[2:0], Blend, 7'b000000000}; // 3 bits
    4'b0100: FormatColor = {Data[3:0], Blend, 6'b000000000}; // 4 bits
    4'b0101: FormatColor = {Data[4:0], Blend, 5'b000000000}; // 5 bits
    4'b0110: FormatColor = {Data[5:0], Blend, 4'b000000000}; // 6 bits
    4'b0111: FormatColor = {Data[6:0], Blend, 3'b000000000}; // 7 bits
    4'b1000: FormatColor = {Data[7:0], Blend, 2'b000000000}; // 8 bits
    4'b1001: FormatColor = {Data[8:0], Blend, 1'b000000000}; // 9 bits
    4'b1010: FormatColor = {Data[9:0], Blend}; // 10
  bits
  default: FormatColor = 'bx;
  andcase
end
endfunction

```

【0203】デプス値の上位16ビットは、DepthMask \* 【0204】  
レジスタフィールドと、ビットごとに論理積をとられ 【数8】  
る。

```

*
function [31:0] FormatDepth;
input [31:0] Data; // Raw depth data
input [15:0] Mask; // Mask
begin
  FormatDepth = {Data[31:16] & Mask,
    Data[15:0]};
end
endfunction

```

【0205】2.2 出力データフォーマット  
オペレーションがRDATである場合、またはオペレーシ  
ョンがRPIXであってDisplayConfigレジスタのPixelFormat  
フィールドが8、16、32ビットピクセルに設定されて  
いる場合、フォーマットはオペレーションのP[3:0]ビ  
ットにしたがって、1024キャッシュラインからデータ  
の64ビットを選択する。ColorWIDUTおよびOverlayWI  
DLUTレジスタはこの場合無視される。

【0206】オペレーションがRPIXであってDisplayCon  
figレジスタのPixelFormatフィールドが64ビットピクセル  
に設定されている場合、フォーマットはオペレーシ  
ョンのP[3:1]ビットによってアドレッシングされた64ビ  
ットピクセルのペアから8ビットWIDフィールドを抽出す  
る。抽出されたWIDフィールドは、ColorWIDUTへのイン  
デックスとなって、色A/Bパッファセレクトのペアが生  
成される。抽出されたWIDフィールドは、OverlayWIDUT  
へのインデックスとなって、16/32ビットセレクト  
のペアが生成される。

【0207】この16/32ビットセレクトは、DQピン  
を介して色データの全32ビットを送信するか、DQピン  
を介して色データの16ビットのみを送信するかを決定  
する。後者の場合、A/Bパッファセレクトが、DQピンを  
介して色データの上位16ビットか下位16ビットのど  
ちらを送信するかを決定する。

【0208】オペレーションがRPIXであって、DisplayC  
onfigレジスタのPixelFormatフィールドが128ビットピ  
クセルに設定されている場合、フォーマットはオペレー  
ションのP[3:2]ビットによってアドレッシングされる12  
8ビットピクセルのペアから8ビットWIDフィールドを  
抽出する。抽出されたWIDフィールドは、ColorWIDUTへ  
のインデックスとなって、色A/Bパッファセレクトのペ  
アが生成される。抽出されたWIDフィールドは、OverlayWI  
DLUTへのインデックスとなって、オーバレイA/Bパッ  
ファセレクトのペアが生成される。

【0209】色A/Bパッファセレクトは、DQピンを介し  
てA色パッファデータかB色パッファデータか、いずれを  
送信するかを決定する。オーバレイA/Bパッファセ  
レクトは、DQピンを介してAまたはBのいずれのオーバレイデ  
ータを送信するかを決定する。

【0210】図68を参照して、このセクションではSR  
AMピクセルパッファ118からレンダリングコントロー  
ラ102へと出ていくデータのルーティングおよび処理  
について説明する。出ていくデータは、SRAM出力デー  
タバス132を介して出力データフォーマット134へと  
送信される。この出力データフォーマット134は、デ  
ュアルピクセル3DRAMチップ110から送信されるピク  
セルフォーマットに応じて、種々のモードで動作する。  
モードは、モードレジスタによって設定される。

【0211】この経路は、RDATおよびRPIXオペレーションによって使用される。ここで、データの256ビットがSRAMビクセルバッファ118から読出され、その256ビットからデータの72ビットまたは108ビットが抽出される。これらはその後、出力データマルチプレクサ136に送られて、チップ110からレンドリグリングコントローラ102へとレンドリグリングバス112を介して送信される。RDATオペレーションは、2サイクル内でデータの64ビットを読出すのに対し、RPIXオペレーションは2サイクルまたは3サイクル内で1ビクセルから8ビクセルを読出す。

【0212】2.2.1 RDAT、RPIX (8ビット、16ビット、32ビットビクセル) オペレーション  
図69は、8ビット、16ビット、および32ビットビクセルフォーマットのためのRDATおよびRPIXオペレーションを図示する。このモードでは、1024ビットキャッシュラインから64ビットが選択される。SRAMビクセルバッファ118が256ビットを提供し、これがマルチプレクサされて64ビットとなる。

【0213】このモードでは、連続する64ビットが1024ビットキャッシュラインから選択されて、出力データマルチプレクサ136に与えられる。

【0214】2.2.2 RPIX (64ビットビクセル) オペレーション  
図71から図74は、64ビットRPIXオペレーションの種々のモードおよび局面を図示する。図71に示したモードにおいては、連続する64ビットのビクセルが2つ、P[0]は無視して、1024ビットキャッシュラインから選択される。

【0215】図72に示したモードでは、P[0]は無視して、2つの連続64ビットビクセルが1024ビットキャッシュラインから選択され、その後処理されて、出力データマルチプレクサ136に提示される。

【0216】偶数の64ビットビクセルは、図73に示すように処理されて、出力データマルチプレクサ136に対して36ビットの出力が生成される。ウィンドウIDビットは、色データが16ビットずつダブルバッファリングするか、32ビットでシングルバッファリングするかを決定し、ダブルバッファリングする場合には、AバッファからBバッファのどちらを選択するかを決定する。ウィンドウIDの8ビットが、256ビットColorWIDULTレジスタおよび256ビットOverlayWIDULTレジスタへのインデックスとなつて、セレクトビットが生成される。

【0217】奇数の64ビットビクセルは、図74に示すように処理されて、出力データマルチプレクサ136に対して36ビットの出力が生成される。ウィンドウIDビットが、色データが16ビットずつダブルバッファリングするか、32ビットでシングルバッファリングするかを決定し、ダブルバッファリングする場合には、Aバ

ッファからBバッファのどちらを選択するかを決定する。

【0218】2.2.3 RPIX (96ビットビクセル) オペレーション

図75から図78は、96ビットRPIXオペレーションの種々のモードおよび局面を図示する。図75に示すように、このモードでは、2つの連続96ビットのビクセルが、P[0]は無視して、1024ビットキャッシュラインから選択される。

【0219】2つの96ビットビクセルは図76に示すように並行に処理されて、出力データマルチプレクサ136に対して2つの48ビット出力が生成される。ウィンドウIDビットは、AバッファからBバッファのどちらを選択するかを決定する。図77および図78は、両ビクセルについてのシングルバッファリングされるオーバーレイおよびウィンドウIDがどのように扱われるかを示す。図78は、色データの低位3バイトのためのデータ経路を示す。

【0220】2.2.4 RPIX (128ビットビクセル) オペレーション

図79から図82は、128ビットビクセルのRPIXオペレーションの種々の局面を図示する。図79に示されたモードにおいて、2つの連続128ビットのビクセルが、P[1:0]は無視して、1024ビットキャッシュラインから選択される。

【0221】図80に示したモードにおいて、2つの連続128ビットビクセルが、P[1:0]は無視して、1024ビットキャッシュラインから選択され、処理されて、出力データマルチプレクサ136に与えられる。

【0222】偶数の128ビットビクセルは、図81に示すように処理されて、出力データマルチプレクサ136に対して48ビット出力が生成される。ウィンドウIDビットは、AまたはBバッファ色、および、AまたはBバッファオーバーレイを選択する。

【0223】奇数の128ビットビクセルは、図82に示されるように処理されて、出力データマルチプレクサ136に対して48ビット出力が生成される。ウィンドウIDビットは、AまたはBバッファ色、および、AまたはBバッファオーバーレイを選択する。

【0224】2.3 SRAMからビクセルALUへのルーティン

図83から図86は、SRAMからビクセルALUへと先行データを送る。SRAMからデータの256ビットを読出して、以下のビクセルALU入力に適切なフィールドを送る。Alpha0[10:0], Red0[10:0], Green0[10:0], Blue0[10:0], Depth0[31:0], Stencil0[7:0], WID0[7:0], Alpha1[10:0], Red1[10:0], Green1[10:0], Blue1[10:0], Depth1[31:0], Stencil1[7:0],およびWID1[7:0]。すべてのビクセルALU入力は、レジスタ値によってマスクされる。

【0225】SRAM→DQおよびSRAM→PALUルーティンブロックは、共通で保有される回路はどれでも共用できる

よう、決して同時に使用されることはない。

【0226】SRAM→PALルーティングは、2段階で行なうことができる。第1段階では、ピクセルのサイズ、アドレス、およびA/B色パッファセレクトにしたがって、各ピクセルの3ビット区分を選択する。第2段階では、A/Bセレクト、色モードおよび種々のマスクにしたがって、色、デプス、ステンシル、およびWIDフィールドをアンパックする。

【0227】2.3.1 8、16、および32ビットピクセルのSRAM編成

これらのピクセルは、アルファ、赤、緑および青データのみを含む。各オペレーションは、SRAMの片方ずつ両方に均等に分割された、64ビット分のピクセルを処理する。

【0228】2.3.2 64ビットピクセルのSRAM編成  
各オペレーションは、SRAMの片方ずつ両方に存在する、2つのピクセルを処理することができる。各ピクセルの「ab」区分は色データを含み、「de」区分はデプス、ス\*

```
function (32:0) UnpackColors;
input (255:0) Data;
input (2:0) PixelSize;
input (3:0) PixelAddress;
input (1:0) BufferSelect;
begin
  case((PixelSize, PixelAddress, BufferSelect[1]))
    // 8, 16, 32 bit pixels
    8'b000_xxx0_x: UnpackColors = { Data[159:128], Data[ 31: 0] }
  }
    8'b000_xxx1_x: UnpackColors = { Data[191:160], Data[ 63:32] }
  }
    8'b000_xxx10_x: UnpackColors = { Data[223:192], Data[ 95:64] }
  }
    8'b000_xxx11_x: UnpackColors = { Data[255:224], Data[127:96] }
  }
    // 64 bit pixels
    8'b001_xxx0_x: UnpackColors = { Data[159:128], Data[ 31: 0] }
  }
    8'b001_xxx1_x: UnpackColors = { Data[223:192], Data[ 95:64] }
  }
    // 96 bit pixels
    8'b010_x00x_0: UnpackColors = { Data[159:128], Data[ 31: 0] }
  }
    8'b010_x00x_1: UnpackColors = { Data[191:160], Data[ 63:32] }
  }
    8'b010_x01x_0: UnpackColors = { Data[159:128], Data[ 95:64] }
  }
    8'b010_x01x_1: UnpackColors = { Data[191:160], Data[127:96] }
  }
    8'b010_x10x_0: UnpackColors = { (32{1'bxx}), Data[ 31: 0] }
  }
    8'b010_x10x_1: UnpackColors = { (32{1'bxx}), Data[ 63:32] }
  }
    // 128 bit pixels
    8'b011_xxxx_0: UnpackColors = { Data[159:128], Data[ 31: 0] }
  }
    8'b011_xxxx_1: UnpackColors = { Data[191:160], Data[ 63:32] }
  }
    default: UnpackColors = 'bx;
  endcase
end
endfunction
```

【0233】2.3.6 UnpackDepths

\* テンシル、オーバーレイ、およびWIDデータを含む。

【0229】2.3.3 96ビットピクセルのSRAM編成  
やはり、各オペレーションは2つのピクセルを処理することができるが、これらの区分はいくぶん交錯している。各ピクセルは、「a」および「b」の個別の色区分を有する。

【0230】2.3.4 128ビットピクセルのSRAM編成  
各オペレーションは、SRAMの片方ずつ両方に存在する2つのピクセルを処理することができる。デプス値は対応する「d」区分にストアされており、ステンシル、オーバーレイ、およびWIDは「e」区分にストアされている。

【0231】2.3.5 UnpackColors  
UnpackColorsは、PixelSize、PixelAddress、およびBufferSelectに基づいて、SRAMから色データの64ビットを選択する。

【0232】

【数9】

50 UnpackDepthsは、PixelSizeおよびPixelAddressに基づ

いて、デプス、ステンシル、オーバーレイ、およびWIDの  
64ビットの値を選択する。 \* 【0234】  
\* 【数10】

```
function (63:0) UnpackDepths;
input [255:0] Data;
input [2:0] PixelSize;
input [3:0] PixelAddress;
begin
  casez([PixelSize, PixelAddress])
    // 64 bit pixels
    7'b001_xxx0: UnpackDepths = { Data[191:160], Data[ 63: 32]
};
    7'b001_xxx1: UnpackDepths = { Data[255:224], Data[127: 96]
};
    // 96 bit pixels
    7'b010_x0xx: UnpackDepths = { Data[223:192], Data[255:224]
};
    7'b010_x10x: UnpackDepths = { {32{1'b0}}, Data[ 95: 64] };
    // 128 bit pixels
    7'b011_xxxx: UnpackDepths = { Data[223:192], Data[ 95: 64]
};
    default: UnpackDepths = 'bx;
  endcase
end
endfunction
```

### 【0235】2.3.7 UnpackExtras

UnpackExtrasは、PixelSizeおよびPixelAddressに基づ  
いて、ステンシル、オーバーレイ、およびWIDの64ビット  
の値を選択する。UnpackDepthsおよびUnpackExtras ※ 【数11】

```
function (63:0) UnpackExtras;
input [255:0] Data;
input [2:0] PixelSize;
input [3:0] PixelAddress;
begin
  casez([PixelSize, PixelAddress])
    // 64 bit pixels
    7'b001_xxx0: UnpackExtras = { Data[191:160], Data[ 63: 32]
};
    7'b001_xxx1: UnpackExtras = { Data[255:224], Data[127: 96]
};
    // 96 bit pixels
    7'b010_x0xx: UnpackExtras = { Data[223:192], Data[255:224]
};
    7'b010_x10x: UnpackExtras = { {32{1'b0}}, Data[ 95: 64] };
    // 128 bit pixels
    7'b011_xxxx: UnpackExtras = { Data[223:192], Data[127: 96]
};
    default: UnpackExtras = 'bx;
  endcase
end
endfunction
```

### 【0237】2.3.8 UnpackAlpha, UnpackRed, UnpackGreen, UnpackBlue

これらの関数は、32ビットの色区分からアルファ、  
赤、緑、および青データをアンパックする。UnpackAlphaはまた、32ビットのエクストラ区分を必要とする。C

20※は、64ビットピクセルおよび96ビットピクセルに  
対して同じデータを返す。

【0236】

【数11】

olorModeおよびBufferSelectは、色のアンパック方法を  
決定する。

【0238】

【数12】

```

67
function [10:0] UnpackAlpha;
    input [31:0] Color;
    input [31:0] Extra;
    input [ 3:0] ColorMode;
    input [ 1:0] BufferSelect;
    begin
        casez({ColorMode, BufferSelect[0]})
            5'b0000_x: UnpackAlpha = {Color[31:24], 3'b100};
            5'b0001_x: UnpackAlpha = {Extra[31:24], 3'b100};
            5'b0010_x: UnpackAlpha = {11{1'b1}};
            5'b0011_0: UnpackAlpha = {Extra[ 7: 0], Color[31:30], 1'b1};
            5'b0011_1: UnpackAlpha = {Extra[15: 8], Color[31:30], 1'b1};
            5'b0100_0: UnpackAlpha = {Color[15:12], 7'b10000000};
            5'b0100_1: UnpackAlpha = {Color[31:28], 7'b10000000};
            5'b0101_0: UnpackAlpha = {Color[15: 8], 3'b100};
            5'b0101_1: UnpackAlpha = {Color[31:24], 3'b100};
            5'b0110_x: UnpackAlpha = {11{1'b1}};
            5'b0111_0: UnpackAlpha = {11{Color[15:1]}};
            5'b0111_1: UnpackAlpha = {11{Color[31:1]}};
            default: UnpackAlpha = 'bx;
        endcase
    end
endfunction

function [10:0] UnpackRed;
    input [31:0] Color;
    input [ 3:0] ColorMode;
    input [ 1:0] BufferSelect;
    begin
        casez({ColorMode, BufferSelect[0]})
            5'b0000_x: UnpackRed = {Color[23:16], 3'b100};
            5'b0001_x: UnpackRed = {11{1'b1}};
            5'b0010_x: UnpackRed = {Color[23:20], 1'b1};
            5'b0100_0: UnpackRed = {Color[11: 8], 7'b10000000};
            5'b0100_1: UnpackRed = {Color[27:24], 7'b10000000};
            5'b0101_0: UnpackRed = {Color[ 7: 0], 3'b100};

```

【0239】

【数13】

(36)

特開 2000-155845

70

69

```

5'b0101_1: UnpackRed = (Color[23:16], 3'b100);
5'b0110_0: UnpackRed = (Color[15:11], 6'b100000);
5'b0110_1: UnpackRed = (Color[31:27], 6'b100000);
5'b0111_0: UnpackRed = (Color[14:10], 6'b100000);
5'b0111_1: UnpackRed = (Color[30:26], 6'b100000);
default: UnpackRed = 'bx;
endcase
end
endfunction

function [10:0] UnpackGreen;
input [31:0] Color;
input [ 3:0] ColorMode;
input [ 1:0] BufferSelect;
begin
case{((ColorMode, BufferSelect[0]))}
5'b0000_x: UnpackGreen = {Color[15: 8], 3'b100};
5'b0001_x: UnpackGreen = {11{1'b1}};
5'b001x_x: UnpackGreen = {Color[19:10], 1'b1};
5'b0100_0: UnpackGreen = {Color[ 4: 7], 7'b1000000};
5'b0100_1: UnpackGreen = {Color[23:20], 7'b1000000};
5'b0110_0: UnpackGreen = {Color[10: 5], 5'b10000};
5'b0110_1: UnpackGreen = {Color[26:21], 5'b10000};
5'b0111_0: UnpackGreen = {Color[ 9: 5], 5'b100000};
5'b0111_1: UnpackGreen = {Color[25:21], 6'b100000};
default: UnpackGreen = 'bx;
endcase
end
endfunction

function [10:0] UnpackBlue;
input [31:0] Color;
input [ 3:0] ColorMode;
input [ 1:0] BufferSelect;
begin
case{((ColorMode, BufferSelect[0]))}
5'b0000_x: UnpackBlue = {Color[ 7: 0], 3'b100};
5'b0001_x: UnpackBlue = {11{1'b1}};
5'b001x_x: UnpackBlue = {Color[ 9: 0], 1'b1};
5'b0100_0: UnpackBlue = {Color[ 3: 0], 7'b1000000};

```

【0240】

【数14】

```

5'b0100_1: UnpackBlue = (Color[19:16], 7'b1000000);
5'b011x_0: UnpackBlue = (Color[ 4: 0], 6'b100000);
5'b011x_1: UnpackBlue = (Color[20:16], 6'b100000);
default: UnpackBlue = 'bx;
endcase
end
endfunction

```

\* 【0241】 2.3.9 UnpackDepth

UnpackDepthは、DepthMaskを使用して、ステンシル、オーバーレイ、およびWIDデータをマスクアウトする。

【0242】

【数15】

```

*40
function [31:0] UnpackDepth;
input [31:0] Depth;
input [15:0] DepthMask;
begin
    UnpackDepth = {(Depth[31:16]&DepthMask),
    Depth[15:0]};
end
endfunction

```

【0243】 2.3.10 UnpackStencil

をマスクアウトする。

UnpackStencilは、StencilMaskを使用してデプスデータ 50

【0244】

(37)

特開2000-155845

71

72

【数16】

```

function [7:0] UnpackStencil;
    input [31:0] Extra;
    input [ 7:0] StencilMask;
    begin
        UnpackStencil = Extra[23:16] & StencilMask;
    end
endfunction

```

【0245】2.3.11 UnpackWid

UnpackWidは、WidMaskを使用してオーバーレイデータをマ  
スクアウトする。

【0246】

【数17】

```

function [7:0] UnpackWid;
    input [31:0] Extra;
    input [ 7:0] WidMask;
    begin
        UnpackWid = Extra[31:24] & WidMask;
    end
endfunction

```

\* 【0247】2.3.12 SramToPaluData

SramToPaluDataは、SRAMデータの256ビットから2つ  
のピクセルについて、32ビットの色、デプス、および  
エクストラ区分を、ならびに、アルファ、赤、緑、青、  
デプス、ステンシル、およびWIDフィールドを、アンバ  
ックする。

【0248】

【数18】

```

function [193:0] SramToPaluData;
    input [255:0] Data;
    input [ 2:0] PixelSize;
    input [ 3:0] PixelAddress;
    input [ 1:0] BufferSelect;
    input [ 3:0] ColorMode;
    input [ 31:0] DepthMask;
    input [ 7:0] StencilMask;
    input [ 7:0] WidMask;

    reg [31:0] Color1, Color0;
    reg [31:0] Depth1, Depth0;
    reg [31:0] Extra1, Extra0;
    begin

        (Color1, Color0) = UnpackColors(Data, PixelSize,
            PixelAddress, BufferSelect);
    end
endfunction

```

【0249】

【数19】



```

(Depth1, Depth0) = UnpackDepths(Data, PixelSize,
    PixelAddress);
(Extra1, Extra0) = UnpackExtras(Data, PixelSize,
    PixelAddress);

SramToFaluData = (
    UnpackWid( Extra1, WidMask),
    UnpackStencil(Extra1, StencilMask),
    UnpackDepth( Depth1, DepthMask),
    UnpackAlpha( Color1, ColorMode,
BufferSelect),
    UnpackRed ( Color1, ColorMode, BufferSelect),
    UnpackGreen( Color1, ColorMode,
BufferSelect),
    UnpackBlue ( Color1, ColorMode,
BufferSelect),

    UnpackWid( Extra0, WidMask),
    UnpackStencil(Extra0, StencilMask),
    UnpackDepth( Depth0, DepthMask),
    UnpackAlpha( Color0, ColorMode,
BufferSelect),
    UnpackRed ( Color0, ColorMode, BufferSelect),
    UnpackGreen( Color0, ColorMode,
BufferSelect),
    UnpackBlue ( Color0, ColorMode, BufferSelect)
);
end
endfunction

```

【0250】2.4 ビクセルALUからSRAMへのデータルー  
 ティング

結果データをビクセルALUからSRAMへと送る。Alpha0[9:0], Red0[9:0], Green0[9:0], blue0[9:0], Depth0[31:0], Stencil0[7:0], DT0, ST0, WT0, Alpha1[9:0], Red1[9:0], Green1[9:0], Blue1[9:0], Depth1[31:0], Stencil1[7:0], DT1, ST1,およびWT1。SRAMに書込まれるデータの各ビットは、対応の書込イネーブルを有する。

【0251】このデータ経路は、ほぼ2つの部分に分割することができる。すなわち、データの32ビットは「0」側の半分から「1」側の半分に送られなければならない。各半分は、ColorPackユニット、DepthPackユニット、およびExtraPackユニットを有する。ColorPackユ

ニットは、ROP/blendユニットの結果を再フォーマットする。DepthPackユニットは、Depth、StencilおよびAlpha ROP/Blendユニットの結果を再フォーマットする。ExtraPackユニットは、AlphaおよびStencilユニットの結果を再フォーマットする。

#### 【0252】2.4.1 PackColor

PackColorは色データの40ビットを入力として、それをColorModeにしたがって32ビットワードにパックする。この関数は、すべてのビクセルサイズで使用される。

【0253】

【数20】

(39)

特開2000-155845

76

```

75
function [31:0] PackColor;
    input [3:0] ColorMode;
    input [9:0] Alpha, Red, Green Blue;
    begin
        case(ColorMode)
            4'd0: PackColor = (Alpha[9:2], Red[9:2], Green[9:2],
Blue[9:2]);
            4'd2: PackColor = { 2'd0, Red[9:0], Green[9:0],
Blue[9:0]};
            4'd3: PackColor = {Alpha[1:0], Red[9:0], Green[9:0],
Blue[9:0]};
            4'd4: PackColor = {2[Alpha[9:6], Red[9:6], Green[9:6],
Blue[9:6]};
            4'd6: PackColor = {2[Red[9:5], Green[9:6], Blue[9:5]};
            4'd7: PackColor = {2[Alpha[9], Red[9:5], Green[9:5],
Blue[9:5]};
        endcase
    end
endfunction

```

#### 【0254】2.4.2 PackDepth

PackDepthは、デプス、ステンシル、およびアルファデータを入力として、それをDepthMaskにしたがって32 \*

\* ビットワードにパックする。

【0255】

【数21】

```

function [31:0] PackDepth;
    input [15:0] DepthMask;
    input [31:0] Depth;
    input [7:0] Stencil;
    input [7:0] Alpha;
    begin
        PackDepth[31:24] = { DepthMask[15:0] & Depth[31:24] } |
(-DepthMask[15:0] & Alpha);
        PackDepth[23:16] = { DepthMask[ 7:0] & Depth[23:16] } |
(-DepthMask[ 7:0] & Stencil);
        PackDepth[15: 0] = Depth[15:0];
    end
endfunction

```

#### 【0256】2.4.3 PackExtra

PackExtraは、ステンシル、およびアルファデータを入力として、それを32ビットワードにパックする。

※【0257】

【数22】

```

function [31:0] PackExtra;
    input [7:0] Stencil;
    input [7:0] Alpha;
    begin
        PackExtra = { Alpha, Stencil, Alpha, Alpha };
    end
endfunction

```

#### 【0258】2.4.4 PaluToSramData

PaluToSramDataは、ピクセルALUの結果を入力として、それをメモリに書込まれるべき256ビットのワードにパックする。PixelSizeおよびPixelAddress入力は、ピクセルフォーマット全体を決定し、一方、ColorMode、A

IphaMaskおよびStencilMaskは、データの詳細なフォーマッティングを決定する。

※【0259】

【数23】

40

```

77
function [255:0] PixelToScanData;
// Per pixel info
input [3:0] PixelAddress;
input [9:0] Alpha0, Red0, Green0, Blue0;
input [9:0] Alpha1, Red1, Green1, Blue1;
input [31:0] DepthL, DepthR;
input [7:0] StencilL, StencilR;
// Register field info
input [2:0] PixelSize;
input [2:0] ColorMode;
input [31:0] DepthMask;

reg [31:0] CPl, CP0, DP1, DP0, EPl, EP0;
reg [255:0] Data;
begin
    CP0 = ColorPack(ColorMode, Alpha0, Red0, Green0, Blue0);
    CP1 = ColorPack(ColorMode, Alpha1, Red1, Green1, Blue1);
    DP0 = DepthPack(DepthMask, Depth0, Stencil0, Alpha0);
    DP1 = DepthPack(DepthMask, Depth1, Stencil1, Alpha1);
    EP0 = { (3{Alpha0}), Stencil0 };
    EP1 = { (3{Alpha1}), Stencil1 };
    case0((PixelSize, PixelAddress))
        7'b000_xxxx: Data = { CPl, CP1, CPl, CP1, CP0, CP0, CP0, CP0 };
        7'b001_xxxx: Data = { DP1, CP1, DP1, CP1, DP0, CP0, DP0, CP0 };
        7'b010_xxxx: Data = { DP0, DP1, CP1, CP1, CP0, CP0, CP0, CP0 };
        7'b010_xxxx: Data = { DP0, DP1, CP1, CP1, CP0, CP0, CP0, CP0 };
        7'b011_xxxx: Data = { EP1, DP1, CP1, CP1, EP0, DP0, CP0, CP0 };
    endcase

    PixelToScanData = Data;
end
endfunction

```

【0260】2.5 ビクセルALUからのマスク生成  
ビクセルALU→SRAMデータ経路のための256ビット書  
込マスクを生成する。

【0261】2.5.1 WriteEnableMask

WriteEnableMaskは、バイト書込イネーブルビットを2

56ビットワードに拡張する。バイト書込イネーブル \*

```

function [255:0] WriteEnableMask;
input [7:0] WriteEnable;
reg [31:0] MaskL, MaskR;
begin
    Mask0 = { (8{WriteEnable[3]}),
    (8{WriteEnable[2]}),
    (8{WriteEnable[1]}), (8{WriteEnable[0]})
};
    Mask1 = { (8{WriteEnable[7]}),
    (8{WriteEnable[6]}),
    (8{WriteEnable[5]}), (8{WriteEnable[4]})
};
    WriteEnableMask = { (4{Mask1}), (4{Mask0}) };
end
endfunction

```

【0263】2.5.2 ビクセルアドレスマスク

32ビットよりも大きいビクセルは、通常、32ビット  
区分へと分割される。PixelAddressMaskは、DualPixe  
l、PixelAddress、およびPixelSizeを使用して、256

\*は、個々の8ビット、16ビット、および32ビットビ  
クセルを選択するのに使用される。これらは8ビット、  
16ビット、および32ビットビクセルについてのみ使  
用されるものである。

【0262】

【数24】

ビットワードのどの32ビット区分に書込がなされるべ  
きかを決定する。

【0264】

【数25】

(41)

特開 2000-155845

80

```

79
function [255:0] PixelAddressMask;
    input      DualPixel;
    input [3:0] PixelAddress;
    input [2:0] PixelSize;
    reg  [1:0] PixelEnable;
    reg  [7:0] WordEnable;
begin
    casex({PixelSize, DualPixel, PixelAddress})
        8'b000_x_x00: WordEnable = 8'b0001_0001; // 8,16,32 bit
        8'b000_x_x01: WordEnable = 8'b0010_0010; // 8,16,32 bit
        8'b000_x_x10: WordEnable = 8'b0100_0100; // 8,16,32 bit
        8'b000_x_x11: WordEnable = 8'b1000_1000; // 8,16,32 bit
        8'b001_0_x00: WordEnable = 8'b0000_0011; // 64 bit
    single
        8'b001_0_x01: WordEnable = 8'b0011_0000; // 64 bit
    single
        8'b001_0_x10: WordEnable = 8'b0000_1100; // 64 bit
    single
        8'b001_0_x11: WordEnable = 8'b1100_0000; // 64 bit
    single
        8'b001_1_x0x: WordEnable = 8'b0011_0011; // 64 bit dual
        8'b001_1_x1x: WordEnable = 8'b1100_1100; // 64 bit dual

        8'b010_0_x00: WordEnable = 8'b1000_0011; // 96 bit
    single
        8'b010_0_x01: WordEnable = 8'b0111_0000; // 96 bit
    single
        8'b010_0_x10: WordEnable = 8'b1000_1100; // 96 bit
    single
        8'b010_0_x11: WordEnable = 8'b0111_0000; // 96 bit
    single
        8'b010_0_x100: WordEnable = 8'b0000_0111; // 96 bit
    single
        * * 【数26】
        8'b010_0_x101: WordEnable = 8'b0000_0000; // 96 bit
    single
        8'b010_0_x11x: WordEnable = 8'b0000_0000; // 96 bit
    single
        8'b010_1_x00x: WordEnable = 8'b1111_0011; // 96 bit dual
        8'b010_1_x01x: WordEnable = 8'b1111_1100; // 96 bit dual
        8'b010_1_x10x: WordEnable = 8'b0000_0111; // 96 bit dual
        8'b010_1_x11x: WordEnable = 8'b0000_0000; // 96 bit dual
        8'b011_0_x00x: WordEnable = 8'b0000_1111; // 128 bit
    single
        8'b011_0_x11x: WordEnable = 8'b1111_0000; // 128 bit
    single
        8'b011_1_xxxx: WordEnable = 8'b1111_1111; // 128 bit
    dual
        endcase

    PixelAddressMask = (
        {32{WordEnable[7]}}, {32{WordEnable[6]}},
        {32{WordEnable[5]}}, {32{WordEnable[4]}},
        {32{WordEnable[3]}}, {32{WordEnable[2]}},
        {32{WordEnable[1]}}, {32{WordEnable[0]}});
    end
endfunction

```

【0265】

81

書込イネーブルに置換されている。

【0267】

\*【数27】

\*

```

function [31:0] MaskDepth
input [15:0] DepthMask;
input      ColorEnable;
input      DepthEnable;
input      StencilEnable;
begin
    MaskDepth[31:24] = (~DepthMask[15:8] & (~DepthEnable)) |
        (~DepthMask[15:8] & (~ColorEnable));
    MaskDepth[23:16] = (~DepthMask[7:0] & (~DepthEnable)) |
        (~DepthMask[7:0] & (~StencilEnable));
    MaskDepth[15:0] = (~DepthMask);
end
endfunction

```

## 【0268】2.5.4 EnableMask

EnableMaskは、ピクセルの色、デプス、およびステンシルフィールドを個別にイネーブル（またはディセーブル）する。ピクセルALUによって行われるWID、ステンシル※

※ルおよびデプステストは、どのフィールドに書込がなされるかを決定する。

【0269】

【数28】

```

function [255:0] EnableMask;
input [31:0] PixelAddress;
input ColorEnable0, ColorEnable1;
input DepthEnable0, DepthEnable1;
input StencilEnable0, StencilEnable1;
input [2:0] PixelSize;
input [31:0] DepthMask;
reg [31:0] CM0, CM1, CM2, CM3, CM4, CM5;
begin
    CM0 = (~ColorEnable0);
    CM1 = (~ColorEnable1);
    CM2 = DepthMask[DepthMask];
    CM3 = DepthMask[DepthMask];
    CM4 = DepthMask[DepthMask];
    CM5 = (~{24(ColorEnable0)}, {8(StencilEnable0)});
    CM1 = (~{24(ColorEnable1)}, {8(StencilEnable1)});

    CaseX[PixelSize, PixelAddress]
    7'b000-xxxx: EnableMask = {CM0, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
    7'b001-xxxx: EnableMask = {CM1, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
    7'b010-xxxx: EnableMask = {CM0, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
    7'b011-xxxx: EnableMask = {CM0, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
    7'b100-xxxx: EnableMask = {CM1, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
    7'b101-xxxx: EnableMask = {CM1, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
    7'b110-xxxx: EnableMask = {CM1, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
    7'b111-xxxx: EnableMask = {CM1, CM1, CM1, CM1, CM0, CM0, CM0,
    CM0};
endcase
end
endfunction

```

## 【0270】2.5.5 SelectPlaneMask

SelectPlaneMaskは、512ビットプレーンマスクレジスタから適切な128ビット区分を選択する。

【0271】

【数29】

```

83
Function [255:0] SelectPlaneMask;
    Input [3:0] PixelAddress;
    Input [2:0] PixelSize;
    Input [511:0] PlaneMask;
    begin
        casez({PixelSize, PixelAddress})
            7'b000-x0xx: SelectPlaneMask = PlaneMask[255:0];
            7'b010-x00x: SelectPlaneMask = PlaneMask[255:0];
            7'b010-x01x: SelectPlaneMask = PlaneMask[511:384];
            PlaneMask[127:0];
            7'b011-x0xx: SelectPlaneMask = PlaneMask[255:0];
            7'b10xx-x0xx: SelectPlaneMask = PlaneMask[255:0];
            7'bxxxx-xxxx: SelectPlaneMask = PlaneMask[511:256];
        endcase
    end
endfunction

```

【0272】2.5.6 ビクセルALUからSRAMへのマスク \* スクを生成する。

PaluToSramMaskは、WriteEnableMask、PixelAddressMask、EnableMask、およびSelectPlaneMaskの結果に対して 【0273】  
【数30】

```

ビットごとに論理積をとって、最後のビットごと書込マスク
Function [255:0] PaluToSramMask;
    // For pixel information
    Input [7:0] WriteEnable;
    Input [0:0] DualPixel;
    Input [3:0] PixelAddress;
    Input [ColorEnable0, ColorEnable1,
    Input [DepthEnable0, DepthEnable1,
    Input [StencilEnable0, StencilEnable1];

    // Info from register fields
    Input [2:0] PixelSize;
    Input [15:0] AlphaMask;
    Input [7:0] StencilMask;
    Input [511:0] PlaneMask;

    begin
        PaluToSramMask =
            WriteEnableMask[WriteEnable] &
            PixelAddressMask[DualPixel, PixelAddress, PixelSize] &
            EnableMask[PixelAddress,
            ColorEnable0, ColorEnable1, DepthEnable0, DepthEnable1,
            StencilEnable0, StencilEnable1, PixelSize, DepthMask] &
            SelectPlaneMask[PixelAddress, PixelSize, PlaneMask];
    end
endfunction

```

【0274】3.0 ビクセルフォーマット

デュアルビクセル3DRAMチップ110は、8ビットから512ビットまでの多種多様のビクセルフォーマットをサポートする。8ビットから32ビットまでのビクセルサイズは、三次元グラフィックスレンダリングオペレーションはサポートしない。256ビットおよび512ビット※

※ビットのビクセルサイズは、マルチサンプリングされたアンチエイリアシングオペレーションをサポートする。下の表2.4は、種々のビクセル速度およびサイズについて、ピークビクセル速度を列挙する。

【表2.4】

表 2.4 種々のビクセル転送およびサイズに対するピークビクセル速度  
(単位: Mビクセル/秒)

ビクセル 当りの ビット数	HL クロック 速度	FL クロック 速度	2D レンダリング 速度	3D レンダリング 速度	デインタール 速度
8	12,800	51,200	1600	—	1600
16	6400	25,600	800	—	800
32	3200	12,800	400	—	400
64	1600	6400	400	267-400	400
128	800	3200	400	200-400	267
256	400	1600	200	66.7	200
512	200	800	200	50	200

【0276】デュアルピクセル3DRAMデバイス110の3つの実施例のピクセル容量を、下の表25に示す。ここで、1Kは1024ビットに等しく、1Mは1024K \*

\* または1,048,576ビットに等しい。

【0277】

【表25】

表25 ピクセルサイズおよびデバイスサイズによるピクセル容量

ピクセル当りの ビット数	40 Mビット	80 Mビット	160 Mビット
8	5,242,880	10,495,760	20,971,520
16	2,621,440	5,242,880	10,485,760
32	1,310,720	2,621,440	5,242,880
64	655,360	1,310,720	2,621,440
128	327,680	655,360	1,310,720
256	163,840	327,680	655,360
512	81,920	163,840	327,680

【0278】図98から図178は、表25に示した種々のピクセルフォーマットがデュアルピクセル3DRAMチップ110においてどのように処理されるかを示す。

【0279】3.1 8ビットピクセルフォーマット

図98から図102は、デュアルピクセル3DRAMチップ110によって8ビットピクセルをどのように処理することができるかを示す。8ビットピクセルでレンダリングされたグラフィックスについては、表示リフレッシュは1バイト/ピクセルを要し、二次元書込は1バイト/ピクセルを要する。

【0280】図98は、8ビットピクセルからライン164へ、およびライン164からページ162への、考えられるディスプレイマッピングを示す。

【0281】図99は、RP1X、SP1XまたはDP1Xオペレーションのための、8ビットピクセルの考えられるキャッシュライン編成を示す。

【0282】図100は、RDATオペレーションを使用する場合に見られる、8ビットピクセルの考えられるキャッシュライン編成を示す。

【0283】図101は、8ビットピクセルのための考えられるフォーマットを示す。ここで、「I」は8ビットインデックスカラーを表わし、これは、SRAMピクセルバッファ118内の256ビットエントリをアドレスリングするのに使用される。各エントリは、赤につき8ビット、緑につき8ビット、および青につき8ビットを有する。これによりプログラムは、ピクセルありわずか8ビットで、可能な16、77、216 (2<sup>8</sup>) 色からどの256色でも選択することができる。

【0284】図102は、図101に特定されたフォーマットの8ビットピクセルの表示リフレッシュオペレーションが8ピクセルを送信するのに2サイクルのRP1Xオペレーションを必要とする様子を示す。

【0285】3.2 16ビットピクセルフォーマット  
図103から図111は、デュアルピクセル3DRAMチップ110によって16ビットピクセルをどのように処理することができるかを示す。二次元ラスタオペレーションは2バイト/ピクセルを要し、二次元ブレンドオペ

レーションは4バイト/ピクセルを要する。

【0286】図103は、16ビットピクセルからライン164へ、およびライン164からページ162への、考えられるディスプレイマッピングを示す。

【0287】図104は、RP1X、SP1XまたはDP1Xオペレーションのための、16ビットピクセルの考えられるキャッシュライン編成を示す。

【0288】図105は、RDATオペレーションを使用する場合に見られる、16ビットピクセルの考えられるキャッシュライン編成を示す。

【0289】図106は、16ビットピクセルのための以下のフォーマットを示す：アルファ：4、赤：4、緑：4、青：4。アルファ、赤、緑および青の各成分は、4ビットで表わされる。

【0290】図107は、図106に特定されたフォーマットの4つの16ビットピクセルを送信するのに、表示リフレッシュオペレーションが2サイクルのRP1Xオペレーションを必要とする様子を示す。

【0291】図108は、16ビットピクセルのための以下の代替的なフォーマットを示す：赤：5、緑：6、青：5。ここで、赤成分は5ビットで表わされ、緑成分は6ビットで、青成分は5ビットで表わされる。このピクセルフォーマットではアルファ成分は使用されない。

【0292】図109は、図108に特定されたフォーマットの4つの16ビットピクセルを送信するのに2サイクルのRP1Xオペレーションを必要とする、表示リフレッシュオペレーションを示す。

【0293】図110は、16ビットピクセルのための以下の代替的なフォーマットを示す：アルファ：1、赤：5、緑：5、青：5。アルファピクセル成分は1ビットで表わされ、赤、緑および青のピクセル成分は各々、5ビットで表わされる。

【0294】図111は、図110に特定されたフォーマットの4つの16ビットピクセルを送信するのに2サイクルのRP1Xオペレーションを必要とする、表示リフレッシュオペレーションを示す。

【0295】3.3 32ビットピクセルフォーマット

図112から図118は、デュアルビクセル3DRAMチップ110によって32ビットビクセルをどのように処理することができるかを示す。表示リフレッシュは4バイト/ビクセルを要し、二次元書込は4バイト/ビクセルを要する。

【0296】図112は、32ビットビクセルからライン164へ、およびライン164からページ162への、考えられるディスプレイマッピングを示す。

【0297】図113は、RPIX、SPIX、またはDPPIXオペレーションのための、32ビットビクセルの考えられるキャッシュライン編成を示す。

【0298】図114は、RDATオペレーションを使用する場合に見られる、32ビットビクセルの考えられるキャッシュライン編成を示す。

【0299】図115は、32ビットビクセルのための以下のフォーマットを示す：アルファ：8、赤：8、緑：8、青：8。アルファ、赤、緑および青のビクセル成分は各々、8ビットで表わされる。

【0300】図116は、図115に特定されたフォーマットの2つの32ビットビクセルを送信するのに、表示リフレッシュオペレーションが2サイクルのRPIXオペレーションを必要とする様子を示す。

【0301】図117は、32ビットビクセルの以下のフォーマットを示す：赤：10、緑：10、青：10。赤、緑および青のビクセル成分は各々、10ビットで表わされる。アルファ成分は存在しない。

【0302】図118は、図117に特定されたフォーマットの2つの32ビットビクセルを送信するのに、表示リフレッシュオペレーションが2サイクルのRPIXオペレーションを必要とする様子を示す。

【0303】3.4 64ビットビクセルフォーマット  
図119から図137は、デュアルビクセル3DRAM110によって64ビットビクセルをどのように処理することができるかを示す。二次元フィルおよび表示リフレッシュオペレーションについては、2つのビクセルは2サイクル内でアクセスすることができる。

【0304】表示リフレッシュは4バイト/ビクセルを要し、二次元レンダリングオペレーションは4バイト/ビクセルを要する。三次元レンダリングオペレーションは、6バイト/ビクセルを要する。

【0305】図119は、64ビットビクセルからライン164へ、およびライン164からページ162への、考えられるディスプレイマッピングを示す。

【0306】図120は、RPIX、SPIX、またはDPPIXオペレーションのための、64ビットビクセルの考えられるキャッシュライン編成を示す。

【0307】図121は、RDATを使用する場合に見られる、64ビットビクセルの考えられるキャッシュライン編成を示す。

【0308】図122は、64ビットビクセルのための

以下のフォーマットを示す：WID：4、アルファ：8、赤：8、緑：8、青：8。アルファ、赤、緑および青のビクセル成分は各々、8ビットで表わされ、ウィンドウIDは4ビットで表わされる。このフォーマットは、三次元のアプリケーションをサポートしない。

【0309】図123は、図122に特定されたフォーマットの2つの64ビットビクセルを送信するのに、表示リフレッシュオペレーションが2サイクルのRPIXオペレーションを必要とする様子を示す。

10 【0310】図124は、64ビットビクセルのための以下のフォーマットを示す：WID：4、赤：10、緑：10、青：10。アルファ、赤、緑および青のビクセル成分は各々、10ビットで表わされる。このフォーマットは、三次元のアプリケーションをサポートしない。

【0311】図125は、図124に特定されたフォーマットの2つの64ビットビクセルを送信するのに、表示リフレッシュオペレーションが2サイクルのRPIXオペレーションを必要とする様子を示す。

20 【0312】図126は、64ビットビクセルのための以下のフォーマットを示す：WID：4、デプス/ステンシル：28、2\*（アルファ：4、赤：4、緑：4、青：4）。このビクセルフォーマットは、各4ビットのアルファ、赤、緑および青のビクセル成分を2組と、28ビットのデプス/ステンシルフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0313】図127は、図126に特定されたフォーマットの2つの64ビットビクセルを送信するのに、表示リフレッシュオペレーションを必要とする、表示リフレッシュシーケンスを示す。

30 【0314】図128は、64ビットビクセルのための以下のフォーマットを示す：WID：4、オーバーレイ：4、デプス/ステンシル：24、2\*（アルファ：4、赤：4、緑：4、青：4）。このビクセルフォーマットは、各々4ビットのアルファ、赤、緑および青のビクセル成分を2組と、24ビットのデプス/ステンシルフィールドと、4ビットのオーバーレイフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0315】図129は、図128に特定されたフォーマットの2つの64ビットビクセルを送信するのに、表示リフレッシュオペレーションが2サイクルのRPIXオペレーションを必要とする様子を示す。

40 【0316】図130は、64ビットビクセルのための以下のフォーマットを示す：WID：4、デプス/ステンシル：28、2\*（赤：5、緑：6、青：5）。このビクセルフォーマットは、5ビットの赤、6ビットの緑および5ビットの青のビクセル成分を2組と、28ビットのデプス/ステンシルフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0317】図131は、図130に特定されたフォーマットの2つの64ビットビクセルを送信するのに、表



示リフレッシュが2サイクルのRPIXオペレーションを必要とする様子を示す。

【0318】図132は、64ビットピクセルのための以下のフォーマットを示す：WID：4、オーパレイ：4、デプス/ステンシル：24、2\*（赤：5、緑：6、青：5）。このピクセルフォーマットは、5ビットの赤、6ビットの緑および5ビットの青のピクセル成分フィールドを2組と、24ビットのデプス/ステンシルフィールドと、4ビットのオーパレイフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0319】図133は、図132に特定されたフォーマットの2つの64ビットピクセルを送信するのに、表示リフレッシュオペレーションが2サイクルのRPIXオペレーションを必要とする様子を示す。

【0320】図134は、64ビットピクセルのための以下のフォーマットを示す：WID：4、デプス/ステンシル：28、2\*（アルファ：1、赤：5、緑：5、青：5）。このフォーマットは、赤、緑および青のピクセル成分につき各々5ビットのフィールドを2組と、アルファピクセル成分のための1ビットフィールドを2組と、28ビットのデプス/ステンシルフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0321】図135は、図134に特定されたフォーマットの2つの64ビットピクセルを送信するのに、表示リフレッシュが2サイクルのRPIXオペレーションを必要とする様子を示す。

【0322】図136は、64ビットピクセルのための以下のフォーマットを示す：WID：4、オーパレイ：4、デプス/ステンシル：24、2\*（アルファ：1、赤：5、緑：5、青：5）。このフォーマットは、赤、緑および青の各ピクセル成分につき各々5ビットのフィールドを2組と、アルファピクセル成分のための1ビットフィールドを2組と、24ビットのデプス/ステンシルフィールドと、4ビットのオーパレイフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0323】図137は、図136に特定されたフォーマットを有する2つの64ビットピクセルを送信するのに、表示リフレッシュが2サイクルのRPIXオペレーションを必要とする様子を示す。

【0324】3.5 96ビットピクセルフォーマット  
図138から図152は、デュアルピクセル3DRAMチップ110によって96ビットピクセルをどのように処理することができるかを示す。このピクセルフォーマットでは、ピクセルあたりデータの56ビットが書込まれ、ピクセルあたり40ビットが表示されて、5ピクセルを8サイクル内にバッキングすることが可能である。表示リフレッシュは6、4バイト/ピクセルを要し、二次元のレンドリングオペレーションは4バイト/ピクセルを要する。

【0325】図138は、96ビットピクセルからライ

ン164へ、およびライン164からページ162への、考えられるディスプレイマッピングを示す。

【0326】図139は、RPIX、SPIX、またはDPPIXオペレーションのための、96ビットピクセルの考えられるキャッシュライン編成を示す。

【0327】図140は、RDATを使用する場合に見られる、96ビットピクセルの考えられるキャッシュライン編成を示す。

【0328】図141は、96ビットピクセルのための1つのフォーマットを示す：WID：4、デプス/ステンシル：28、2\*（オーパレイ：8、赤：8、緑：8、青：8）。このフォーマットは、オーパレイ、赤、緑および青の各ピクセル成分につき各々8ビットのフィールドを2組と、28ビットのデプス/ステンシルフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0329】図142は、表示リフレッシュが、図141に特定されたフォーマットを有する2つの96ビットピクセルを送信するのに3サイクルのRPIXオペレーションを、または、図141に特定されたフォーマットを有する1つの96ビットピクセルを送信するのに2サイクルのRPIXオペレーションを、必要とする様子を示す。

【0330】図143は、96ビットピクセルのための以下のフォーマットを示す：WID：4、デプス/ステンシル：28、2\*（アルファ：8、赤：8、緑：8、青：8）。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々8ビットのフィールドを2組と、28ビットのデプス/ステンシルフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0331】図144は、図143に特定された96ビットピクセルフォーマットの表示リフレッシュが、2つのピクセルを送信するのに3サイクルのRPIXオペレーションを、または、1つのピクセルを送信するのに2サイクルのRPIXオペレーションを必要とする様子を示す。

【0332】図145は、96ビットピクセルのための以下のフォーマットを示す：WID：4、オーパレイ：4、デプス/ステンシル：24、2\*（アルファ：8、赤：8、緑：8、青：8）。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々8ビットのフィールドを2組と、4ビットのオーパレイフィールドと、4ビットのウィンドウIDフィールドとを含む。

【0333】図146は、図145に特定された96ビットピクセルフォーマットの表示リフレッシュが、2つのピクセルを送信するのに3サイクルのRPIXオペレーションを、または、1つのピクセルを送信するのに2サイクルのRPIXオペレーションを必要とする様子を示す。

【0334】図147は、96ビットピクセルのための以下のフォーマットを示す：WID：4、デプス/ステンシル：28、2\*（赤：10、緑：10、青：10）。このフォーマットは、赤、緑および青の各ピクセル成分

10

20

30

40

50

につき各々 10 ビットのフィールドを 2 組と、28 ビットのデプス/ステンシルフィールドと、4 ビットのウィンドウ ID フィールドとを含む。

【0335】図 148 は、図 147 に特定された 96 ビットピクセルフォーマットの表示リフレッシュが、2 つのピクセルを送信するのに 3 サイクルの RPIX オペレーションを、または、1 つのピクセルを送信するのに 2 サイクルの RPIX オペレーションを必要とする様子を示す。

【0336】図 149 は、96 ビットピクセルのための以下のフォーマットを示す：WID: 4、オーバレイ: 4、デプス/ステンシル: 24、2\* (赤: 10、緑: 10、青: 10)。このフォーマットは、赤、緑および青の各ピクセル成分につき各々 10 ビットのフィールドを 2 組と、24 ビットのデプス/ステンシルフィールドと、4 ビットのオーバレイフィールドと、4 ビットのウィンドウ ID フィールドとを含む。

【0337】図 150 は、図 149 に特定された 96 ビットピクセルフォーマットの表示リフレッシュが、2 つのピクセルを送信するのに 3 サイクルの RPIX オペレーションを、または、1 つのピクセルを送信するのに 2 サイクルの RPIX オペレーションを必要とする様子を示す。

【0338】図 151 は、96 ビットピクセルのための以下のフォーマットを示す：WID: 4、オーバレイ: 4、デプス/ステンシル: 24、4\* (アルファ: 4、赤: 4、緑: 4、青: 4)。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々 4 ビットのフィールドを 2 組と、24 ビットのデプス/ステンシルフィールドと、4 ビットのオーバレイフィールドと、4 ビットのウィンドウ ID フィールドとを含む。

【0339】図 152 は、図 151 に特定された 96 ビットピクセルフォーマットの表示リフレッシュが、2 つのピクセルを送信するのに 3 サイクルの RPIX オペレーションを、または、1 つのピクセルを送信するのに 2 サイクルの RPIX オペレーションを必要とする様子を示す。

【0340】3.6 128 ビットピクセルフォーマット  
図 153 から図 161 は、デュアルピクセル 3D RAM チップ 11.0 によって 128 ビットピクセルをどのように処理することができるかを示す。このデプスにおいては、ピクセルあたり 64 ビットが書込まれ、ピクセルあたり 48 ビットが表示されて、4 ピクセルを 3 サイクル内にパッキングすることが可能となる。2 次元の性能を高めるために、ピクセルをサイクルあたり 2 ピクセルの速度で更新することもできる。表示リフレッシュは 6 バイト/ピクセルを要し、2 次元のレンダリングオペレーションは 4 バイト/ピクセルを要する。128 ビットピクセルフォーマットはすべて、一定位置の 8 ビット WID フィールドを有する。

【0341】図 153 は、128 ビットピクセルからライン 164 へ、およびライン 164 からページ 162 への、考えられるディスプレイマッピングを示す。

【0342】図 154 は、RPIX、SPIX、または DPPIX オペレーションのための、128 ビットピクセルの考えられるキャッシュライン編成を示す。

【0343】図 155 は、RDAT オペレーションを使用する場合に見られる、128 ビットピクセルの考えられるキャッシュライン編成を示す。

【0344】図 156 は、128 ビットピクセルのための以下のフォーマットを示す：WID: 8、デプス: 32、ステンシル: 8、2\* (オーバレイ: 8、アルファ: 8、赤: 8、緑: 8、青: 8)。このフォーマットは、オーバレイ、アルファ、赤、緑および青の各ピクセル成分につき各々 8 ビットのフィールドを 2 組と、8 ビットのステンシルフィールドと、32 ビットのデプスフィールドと、8 ビットのウィンドウ ID フィールドとを含む。

【0345】図 157 は、図 156 に特定された 128 ビットピクセルフォーマットの表示リフレッシュが、2 つのピクセルを送信するのに 3 サイクルの RPIX オペレーションを必要とする様子を示す。

【0346】図 158 は、128 ビットピクセルのための以下のフォーマットを示す：WID: 8、デプス: 32、ステンシル: 8、2\* (オーバレイ: 8、赤: 10、緑: 10、青: 10)。このフォーマットは、赤、緑、および青の各ピクセル成分につき各々 10 ビットのフィールドを 2 組と、オーバレイのための 8 ビットのフィールドを 2 組と、8 ビットのステンシルフィールドと、32 ビットのデプスフィールドと、8 ビットのウィンドウ ID フィールドとを含む。

【0347】図 159 は、図 158 に特定されたフォーマットを有する 128 ビットピクセルの表示リフレッシュが、2 つのピクセルを送信するのに 3 サイクルの RPIX オペレーションを必要とする様子を示す。

【0348】図 160 は、128 ビットピクセルのための以下のフォーマットを示す：WID: 8、ステンシル: 8、デプス: 32、2\* (アルファ: 10、赤: 10、緑: 10、青: 10)。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々 10 ビットのフィールドを 2 組と、32 ビットのデプスフィールドと、8 ビットのステンシルフィールドと、8 ビットのウィンドウ ID フィールドとを含む。

【0349】図 161 は、図 160 に特定されたフォーマットを有する 128 ビットピクセルの表示リフレッシュが、2 つのピクセルを送信するのに 3 サイクルの RPIX オペレーションを必要とする様子を示す。

【0350】3.7 マルチサンプル・ポリゴン・アンチエイリアシング

256 ビットまたは 512 ビットのフォーマットを有するピクセルは、マルチサンプル・ポリゴン・アンチエイリアシング・レンダリングオペレーションをサポートする。いくつかのポリゴン・アンチエイリアシングのアル

ゴリズムを、それらに関連する利点に焦点を当てて、以下に説明する。

### 【0351】 3.7.1 累算バッファ

このアルゴリズムは、複数のエイリアシングされたイメージを累算して、最終のアンチエイリアシングされたイメージを生成する。エイリアシングされた各イメージは、XおよびY方向にオフセットされた異なるサブピクセルでレンダリングされる。エイリアシングされた最終イメージは、累算バッファの現時点における内容に累算される。すべてのエイリアシングされたイメージがレンダリングされかつ累算されると、累算バッファ内のピクセル成分を累算されたイメージの数で除して、最終的なアンチエイリアシングされたイメージが生成される。

【0352】 累算バッファは、非常に柔軟な技術であって、ポリゴン・アンチエイリアシングに加えて、モーションブラー、フィールドの奥行き、ソフトシャドウに対応することができる。

【0353】 累算バッファは品質および柔軟性の面で非常に優れているが、この技術はアンチエイリアシングされた最終的なイメージを生成するのに時間がかかりすぎるため、高いフレームレートのアプリケーションには不適切な場合がある。

【0354】 この技術に関する詳細な説明は、以下の出版物に記載されており、これをここに引用により援用する。ポール・ヘバーリ (Haerberli, Paul)、K. アケレー (K. Akeley) による「累算バッファ：高品質レンダリングのためのハードウェアサポート ("The Accumulation Buffer: Hardware Support for High-Quality Rendering")」、Computer Graphics, Vol. 24, No. 4, 1990年8月、第309～318頁。

### 【0355】 3.7.2 Aバッファ

このアルゴリズムは、各ピクセルについて、ポリゴンフラグメントのソートされたリストを保持する。もし1つの三角形が1つのピクセルを完全にカバーし、かつ完全に不透明である場合、その三角形の背後のフラグメントはすべて捨ててもよい。そうでない場合には、フラグメントはそのリストの適切な場所に挿入される。各フラグメントは、最小でも、色、デプス、ピクセルマスク、および次のフラグメントへのポインタ、の成分を有する。フレーム全体のレンダリングが終わった後に、すべてのピクセルについて最終的な色へのフラグメントリストを決めるのに、付加的な処理が必要である。

【0356】 Aバッファのアルゴリズムは、透明な三角形を正しくレンダリングするのに非常に優れており、それらの三角形がデプスによってソートされていない場合であっても問題はない。Aバッファのアルゴリズムは通常、三角形が互いに交わる場合には、相当量の付加的な情報が各フラグメントにストアされていない限り、うまく処理することができる。Aバッファアルゴリズムの主要な欠点は、ピクセルあたり、無限量の記憶領域および

処理を要することである。必要とされるフレームバッファ記憶領域は、1フレーム内のポリゴンの数にほぼ比例する。

【0357】 この技術の詳細な説明は、以下の出版物に記載されており、これをここに引用により援用する。ローレン・カーペンター (Carpenter, Loren) による「A バッファ、アンチエイリアス隠面方法 ("The A-buffer, an Anti-Aliased Hidden Surface Method")」、Computer Graphics, Vol. 18, No. 3, 1984年7月、第103～108頁。

### 【0358】 3.7.3 マルチサンプル

マルチサンプルアンチエイリアシングは、ピクセルあたりいくつかのサンプルをストアする。各サンプルは、そのピクセル内またはそのピクセルの近辺の異なる場所に位置決めされる。サンプルは、色情報をストアするか、または、色、デプスおよびステンスルの情報をストアする。ピクセルをレンダリングするとき、三角形の内部に存在するサンプルが算出されて、フレームバッファ内のサンプルとマージされる。フレーム全体がレンダリングされた後に、サンプルのすべての色の重み付平均が、表示装置に送られる。

【0359】 図162は、2つのピクセルおよび、各ピクセル内の各サンプルの場所を示す。現時点の三角形の内部のサンプルは中黒の丸で示し、その三角形の外部のサンプルは中空の丸で示す。

【0360】 マルチサンプルアンチエイリアシングは、フレームあたりレンダリングされる三角形の数にかかわらず、ピクセルあたり一定量の記憶領域を必要とする。この技術は、OpenGLまたはDirectXに対するAPIの変更をほとんど必要とせず、必要なのは、アンチエイリアシング機能をイネーブルまたはディセーブルするための何らかの方法のみである。

【0361】 この技術の詳細な説明は以下の出版物に記載されており、これをここに引用により援用する。カート・アケレー (Akeley, Kurt) による「リアリティエンジン・グラフィックス ("Reality Engine Graphics")」、Computer Graphics, 1993年8月、第109～116頁。

【0362】 3.7.3.1 サンプルあたり色のみの場合、各ピクセルはWD、ステンスル、デプス、バックカラー、フロントカラーのための記憶領域を有し、各サンプルは色のための記憶領域を有する。レンダリングコントロール102は、共通のピクセル色、共通のピクセルデプス、および、そのサンプルが三角形の内部にあるか否かを示すサンプルあたり1ビットを送る。三角形の辺は、正しくアンチエイリアシングされる。ただし、三角形の交点は、ピクセルがデプス値を1つしかストアしていないため、エイリアシングされるであろう。これを下記、verilogコードで説明する。

### 【0363】

【数31】

```

if WID test passes {
    update stencil
    if stencil and depth tests pass {
        colorsum = 0
        for each sample {
            if sample is inside triangle
                merge source color with sample's color
                colorsum. += sample's color
        }
        back color = colorsum / # of samples
        overwrite depth
    }
}

```

【0364】3.7.3.2 サンプルあたり色およびデプス  
この場合、各ピクセルはWID、バックカラー、およびフ  
ロントカラーのための記憶領域を有し、各サンプルは  
色、デプス、およびステンシルのための記憶領域を有す  
る。レンダリングコントローラ102は、共通のピクセ  
ル色、サンプルあたり1つの異なるデプス値、および、\*

\* そのサンプルが三角形の内部にあるか否かを示すサン  
プルあたり1ビットを送る。三角形の辺および交点の両方  
が、正しくアンチエイリアシングされるであろう。これ  
を下に、verilogコードで説明する。

【0365】

【数32】

```

if WID test passes {
    colorsum = 0
    for each sample {
        if sample is inside triangle {
            update sample stencil
            if sample's stencil and depth tests pass {
                merge source color with sample color
                overwrite sample depth
            }
            colorsum += sample color
        }
    }
    back color = colorsum / # of samples
}

```

【0366】3.7.4 サンプルあたり色およびデプスの速  
度改良

デプス値は通常、次のように、XおよびYの1次関数とし  
て計算される。

【0367】

【数33】

$$\text{Depth}(X, Y) = \frac{d\text{Depth}}{dX} X + \frac{d\text{Depth}}{dY} Y + \text{Depth}(0, 0) \quad 40$$

※

$$\text{Depth}(X + \Delta X, Y + \Delta Y) \approx \text{Depth}(X, Y) + \frac{d\text{Depth}}{dX} \Delta X + \frac{d\text{Depth}}{dY} \Delta Y$$

【0370】通常、ピクセル内のサンプル位置は、図1  
63に示すように、ピクセル内の中央、隅部、またはあ  
る固定された基準点から、XおよびY方向に同じだけオフ  
セットされている。

【0371】もし、ピクセルの中央（または他の何らか

【0368】この等式は下のように書き換えることがで  
きるが、この方がより有益である。

【0369】

【数34】

の基準点)におけるデプスがわかっていれば、サンプル  
nのデプスは、次の式で計算することができる。

【0372】

【数35】

$$\text{Depth}(X + \Delta X_n, Y + \Delta Y_n) = \text{Depth}(X, Y) + \frac{d\text{Depth}}{dX} \Delta X_n + \frac{d\text{Depth}}{dY} \Delta Y_n$$

【0373】この計算は、2つのステップで行なうこと \* 【0374】  
 ができる。 【数36】

$$\Delta \text{Depth}_n = \frac{d\text{Depth}}{dX} \Delta X_n + \frac{d\text{Depth}}{dY} \Delta Y_n$$

$$\text{Depth}(X + \Delta X_n, Y + \Delta Y_n) = \text{Depth}(X, Y) + \Delta \text{Depth}_n$$

ここで、 $\frac{d\text{Depth}}{dX}$  および  $\frac{d\text{Depth}}{dY}$  の項は、同じ三角形の中すべての

ピクセルについて同じである。したがって、 $\Delta \text{Depth}_n$  の項もまた、

同じ三角形内のすべてのピクセルについて同じとなる。

【0375】1つの三角形につき、各サンプルの (Depth, X, Y) 項をすべて一度に計算して、それらをレジスタ書込としてデュアルピクセル3DRAMのピクセルALUへと、その三角形内の最初のピクセルをレンダリングする前に、送信することが可能である。その後、その三角形内の各ピクセルについて、ピクセルの中央 (または何らかの他の基準点) における色およびデプス値のみが送信される。各サンプルにおけるデプス値は、以下の式を用いて計算される。

【0376】

【数37】

$$\text{Depth}(X + \Delta X_n, Y + \Delta Y_n) = \text{Depth}(X, Y) + \Delta \text{Depth}_n$$

【0377】もし各サンプルがデュアルピクセル3DRAMのピクセルALU内に専用のデプス加算器、デプス比較器、および色ブレンドユニットを有する場合には、アンチエイリアシングされたピクセル全体を1つのオペレーションでレンダリングすることが可能であろう。デュアルピクセル3DRAMチップ110は、このような能力を有する。

【0378】3.8 256ビットピクセルフォーマット (4×マルチサンプル)

図164から図169は、デュアルピクセル3DRAMチップ110によって256ビットピクセルをどのように処理することができるかを示す。128ビットピクセルに加えて、チップ110は、ピクセルあたり4つのサブサンプルをストアすることができる。ピクセルを書込むために、チップ110は色およびデプス値、ならびに、4ビットのサンプルマスクを送る。ピクセルALUがサイクルあたり2つのサンプルを処理することができる場合、チップ110は2サイクルで各ピクセルを更新することが可能である。ピクセルが更新される間、4つのサンプルのすべての色値が累算されて、または8色バッファに書込まれる。

【0379】マルチサンプルアンチエイリアシングは、Aバッファをベースとするアンチエイリアシングよりも実装が容易である。マルチサンプルは、ピクセルを更新するのに、ピクセルあたり大量ではあるが有限量の記憶

領域と、一定量の時間とを要するが、フレームのレンダリングとそのフレームの表示との間に、フラグメント決定処理ステージを必要としない。三角形の速度が1秒あたり数百万個であるとき、ほとんどのピクセルは部分的にカバーされており、Aバッファのフィル速度は、フラグメントの記憶領域の要求が飛躍的に増大するのにつれて低速化するのである。

【0380】表示リフレッシュは8バイト/ピクセルを要し、二次元のレンダリングは4バイト/ピクセルを、三次元のレンダリングは9バイト/ピクセルを要する。

【0381】256ビットピクセルフォーマットはすべて、一定位置の8ビットWIDフィールドを有する。すべてのレンダリングフォーマットは、2サイクルのSPIXオペレーションまたは2サイクルのDPIXオペレーションを要する。表示リフレッシュは、1つのピクセルを送信するのに、2サイクルのRPIXオペレーションを必要とする。

【0382】図164は、256ビットピクセルからライン164へ、および、ライン164からページ162への、考えられるディスプレイマッピングを示す。

【0383】図165は、RPIX、SPIX、およびDPIXオペレーションのための、256ビットピクセルの考えられるキャッシュライン編成を示す。

【0384】図166は、RDATオペレーションを使用する場合に見られる、256ビットピクセルの考えられるキャッシュライン編成を示す。

【0385】図167は、256ビットピクセルのための以下のフォーマットを示す：WID：8、ステンシル：8、デプス：32、2\* (オーバーレイ：8、アルファ：8、赤：8、緑：8、青：8)、4\* (アルファ：8、赤：8、緑：8、青：8)。このフォーマットは、ウィンドウIDおよびステンシルにつき各8ビットのフィールドと、デプスのための32ビットのフィールドと、オーバーレイ、アルファ、赤、緑および青の各ピクセル成分につき各々8ビットのフィールドを2組と、さらに、アルファ、赤、緑および青の各8ビットの成分を含む4つの色サンプルとを含む。

【0386】図168は、図167に特定された256ビットピクセルフォーマットの表示リフレッシュが、1つのピクセルを送信するのに2サイクルのRPIXオペレーションを必要とする様子を示す。

【0387】図169は、256ビットピクセルフォーマットのための、SRAM読出/書込フォーマットを示す。

【0388】3.9 512ビットピクセルフォーマット(6マルチサンプル)

図170から図178は、デュアルピクセル3DRAMチップ110によって512ビットピクセルをどのようにに処理することができるかを示す。このピクセルフォーマットは、6マルチサンプリングオペレーションをサポートする。このフォーマットにおいて、ピクセルあたり6つのサブサンプルが、128ビットピクセルに加えてストアされる。ピクセルを書込むために、共通の色値が最初に送信され、これに6つのデプス値が続く。もしピクセルALUが1サイクルあたり2つのサンプルを処理することができれば、各ピクセルは8サイクルで更新することが可能である。ピクセルが更新されている間に、6つのサンプルのすべての色値が累算されて、AまたはB色バッファに書込まれる。

【0389】マルチサンプリングによるアンチエイリアシングは、Aバッファをベースとするアンチエイリアシングよりも実装が容易である。マルチサンプリングは、ピクセルを更新するのに、ピクセルあたり大量ではあるが有限量の記憶領域、および、一定量の時間を必要とするが、フレームのレンダリングとそのフレームの表示との間に、フラグメント決定ステージを必要としない。このマルチサンプリングの技術は、互いに貫通する面をアンチエイリアシングする。Aバッファを強化して互いに貫通する面をアンチエイリアシングするのは費用が高つく。三角形の速度が1秒あたり数百万個である場合、ほとんどのピクセルは部分的にカバーされており、Aバッファのフィル速度は、フラグメント記憶領域の要件が大いに増大する一方で、低速化することになる。

【0390】表示リフレッシュは8バイト/ピクセルを要し、二次元のレンダリングオペレーションは4バイト/ピクセルを、三次元のレンダリングオペレーションは32バイト/ピクセルを要する。

【0391】512ビットピクセルフォーマットはすべて、一定位置の8ビットWIDフィールドを有する。

【0392】図170は、512ビットピクセルからライン164へ、および、ライン164からページ162への、考えられるディスプレイマッピングを示す。

【0393】図171は、RPIX、SPIXまたはADPIXオペレーションのための、512ビットピクセルの考えられるキャッシュライン編成を示す。

【0394】図172は、RDATオペレーションを使用する場合に見られる、512ビットピクセルの考えられるキャッシュライン編成を示す。

【0395】図173は、512ビットピクセルのための1つのフォーマットを示す：WID: 8、2\* (オーバーレイ: 8、アルファ: 8、赤: 8、緑: 8、青: 8)、6\* (アルファ: 8、赤: 8、緑: 8、青: 8、ステンシル/デプス: 32)。このフォーマットは、各サンプルがアルファ、赤、緑および青の各ピクセル成分につき各々8ビットのフィールドおよびデプス/ステンシルのための32ビットのフィールドを有する6つのピクセルサンプルと、オーバーレイ、アルファ、赤、緑および青の各ピクセル成分につき各々8ビットのフィールドを2組と、8ビットのウィンドウIDを含む。

【0396】図174は、図173に特定された512ビットピクセルフォーマットの表示リフレッシュが、1つのピクセルを送信するのに2サイクルのRPIXオペレーションを必要とする様子を示す。

【0397】図175は、図173に特定されたフォーマットの512ビットピクセルのための、SRAMピクセルバッファ118への読出フォーマットおよびSRAMピクセルバッファ118からの書込フォーマットを示す。

【0398】図176は、512ビットピクセルのための以下のフォーマットを示す：WID: 8、2\* (オーバーレイ: 8、赤: 10、緑: 10、青: 10)、6\* (赤: 10、緑: 10、青: 10、ステンシル/デプス: 32)。このフォーマットは、各サンプルが赤、緑および青の各ピクセル成分につき各々10ビットのフィールドおよびデプス/ステンシルのための32ビットのフィールドを有する6つのピクセルサンプルと、赤、緑および青の各ピクセル成分につき各々10ビットのフィールドを2組と、オーバーレイのための8ビットのフィールドを2組と、8ビットのウィンドウIDを含む。

【0399】図177は、図176に特定された512ビットピクセルフォーマットの表示リフレッシュが、1つのピクセルを送信するのに2サイクルのRPIXオペレーションを必要とする様子を示す。

【0400】図178は、図176に特定されたフォーマットの512ビットピクセルのための、SRAMピクセルバッファ118への読出フォーマットおよびSRAMピクセルバッファ118からの書込フォーマットを示す。

【0401】4.0 双方向I/O  
高速かつ同時の双方向送受信の実行可能ないくつかの実装例が、最近提示されてきている。この技術は、2つのデバイスをガガビット速度で接続する単一の配線を通して、データを双方向で送信することを可能にする。このセクションでは、このI/O技術を性能の向上またはコストの低減のためにデュアルピクセル3DRAMデバイスにどのように適用することができるかを示す。開示される高速かつ同時の双方向送受信の詳細な説明は、以下の出版物に提示されており、これに引用により援用する。M.ヘイコック (Haycock, M.)、R.ムーニー (Mooney, R.) による「2.5Gb/s双方向送受信技術 (A 2.5

Gb/s Bidirectional Signaling Technology)」、Hot Interconnects Symposium V、1997年8月、第149～156頁。

【0402】図179は、データピンを時分割するのに代えて、制御/アドレスピンを介して表示リフレッシュデータを送ることによって、性能を向上させる方式を図示する。この表示リフレッシュ経路は、表示リフレッシュ情報をフェッチするために制御およびアドレス情報を生成するのに、スタートマシンを必要とする。データピン上のトラフィックは通常、単一方向である。

【0403】図180は、ピクセルALUをレンダリングコントロールへと移動させることを可能にする方式を図示する。これらデータピンは、ソースおよび結果ピクセルデータの同時送信を可能にする。制御/アドレス情報および表示リフレッシュデータは、ピンの同じ組を共有する。

#### 【図面の簡単な説明】

【図1】1組のデュアルピクセル3DRAMチップと、レンダリングコントロールと、レンダリングバスとアドレスおよび制御バスとの対と、ビデオ出力回路を含む、ビデオディスプレイフレームバッファとも称されるグラフィックスサブシステムを示すブロック図である。

【図2】DRAMレイ、SRAMピクセルバッファ、2つのピクセル算術論理演算装置 (ALU) およびグローバルバスを含むデュアルピクセル3DRAMチップの一実施例を示す機能ブロック図である。

【図3】RAMBUS™ またはSyncLink入力/出力インタフェース仕様のいずれかで動作するよう構成される、図2に示される要素を含むデュアルピクセル3DRAMチップの一実施例のためのダイサイズフロアプランの図である。

【図4】いくつかの処理要素の回路のある部分が共有される、2組の処理要素を含むピクセルALUの一実施例のブロック図である。

【図5】2つの別個の組の処理要素を含むピクセルALUの一実施例のブロック図である。

【図6】ピクセルALUの一実施例のための、4つのラストオペレーション (ROP) /ブレンドユニット (ピクセルのアルファ、赤、緑および青成分の各々を処理するために1つずつ) と、デプスユニットと、ステンシルユニットと、ウィンドウ識別 (ID) ユニットとを含む処理要素の完全な組と、これらの処理要素に入力され、これらの処理要素から出力される情報の種類とを示す図である。

【図7】1つのROP/ブレンドユニットの一実施例を示すブロック図である。

【図8】1つのラストオペレーション (ROP) ユニットの一実施例を示すブロック図である。

【図9】1つの8ビットブレンドユニットの一実施例を示すブロック図である。

【図10】1組の8ビットブレンドユニット計算を示す

図である。

【図11】ブレンドユニットにおいて用いられるドット分散型組織的ディザルゴリズムを示す図である。

【図12】10ビットブレンドユニットの一実施例を示すブロック図である。

【図13】1組の10ビットブレンドユニット計算を示す図である。

【図14】一実施例のためのROP/ブレンドユニットへの入力マルチプレクサの詳細を示す図である。

10 【図15】デプス比較ユニットの一実施例を示すブロック図である。

【図16】ステンシル比較ユニットの一実施例を示すブロック図である。

【図17】ステンシルデータ経路の一実施例を示すブロック図である。

【図18】ウィンドウID比較ユニットの一実施例を示すブロック図である。

【図19】SRAMピクセルバッファの一実施例を示すブロック図である。

20 【図20】SRAMピクセルバッファの第2の実施例を示すブロック図である。

【図21】高レベルでのデュアルピクセル3DRAMチップのメモリ構成を示す図である。

【図22】デュアルピクセル3DRAMチップのメモリ構成を示すより詳細なブロック図である。

【図23】SRAMピクセルバッファに関連してDRAMバンクメモリ構成の一実施例を示すブロック図である。

【図24】アドレスおよび制御ピンを示すブロック図である。

30 【図25】アドレスおよび制御 (RQ) ピンを介するデュアルピクセル3DRAMプロトコル構造を示す図である。

【図26】アドレスおよび制御ピンを介するバンクオペレーションプロトコルを示す図である。

【図27】ページリチャージオペレーションを示すタイミングダイアグラムの図である。

【図28】ページアクセスオペレーションを示すタイミングダイアグラムの図である。

【図29】ページ変更オペレーションを示すタイミングダイアグラムの図である。

40 【図30】アドレスおよび制御 (RQ) ピンを介するアイドルコマンドを示す図である。

【図31】アドレスおよび制御 (RQ) ピンを介するRead Cache Line (RL) コマンド、Write Cache Line (WL) コマンド、Masked Write Cache Line (ML) コマンドおよびChange Cache Line (CL) コマンドを示す図である。

【図32】アドレスおよび制御 (RQ) ピンを介するFlash Masked Write Cache Line コマンドを示す図である。

【図33】Read Cache Line (RL) オペレーションを示すタイミングダイアグラムの図である。

【図 3 4】 Write Cache Line (WL) オペレーションを示すタイミングダイアグラムの図である。

【図 3 5】 Masked Write Cache Line (ML) オペレーションを示すタイミングダイアグラムの図である。

【図 3 6】 Flash Masked Write Cache Line (FL) オペレーションを示すタイミングダイアグラムの図である。

【図 3 7】 Change Cache Line (CL) オペレーションを示すタイミングダイアグラムの図である。

【図 3 8】 アドレスおよび制御 (RQ) ピンを介するアイドルコマンドを示す図である。

【図 3 9】 アドレスおよび制御 (RQ) ピンを介する Read Data (RDAT) コマンド、Write Data (WDAT) コマンドおよび Broadcast Data (BDAT) コマンドを示す図である。

【図 4 0】 アドレスおよび制御ピンを介する Read Register (RREG) コマンド、Write Register (WREG) コマンドおよび Broadcast Register (BREG) コマンドを示す図である。

【図 4 1】 アドレスおよび制御ピンを介する Read Pixel (RPIX) コマンド、Single Pixel (SPIX) コマンドおよび Dual Pixel (DPIX) コマンドを示す図である。

【図 4 2】 データ (DQ) ピンを介するグラフィックスデータ転送を示す図である。

【図 4 3】 2 サイクル RDAT オペレーションとその後の 3 サイクル RPIX オペレーションを示すタイミングダイアグラムの図である。

【図 4 4】 WDAT、BDAT、WREG および DREG オペレーションのためのタイミングダイアグラムの図である。

【図 4 5】 SPIX および DPIX オペレーションのためのタイミングダイアグラムの図である。

【図 4 6】 3 サイクル RPIX 転送オペレーションを示すタイミングダイアグラムの図である。

【図 4 7】 複合 2 サイクル読出および 2 サイクル書込オペレーションを示すタイミングダイアグラムの図である。

【図 4 8】 複合 2 サイクル読出および 3 サイクル書込オペレーションを示すタイミングダイアグラムの図である。

【図 4 9】 複合 3 サイクル読出および 2 サイクル書込オペレーションを示すタイミングダイアグラムの図である。

【図 5 0】 複合 3 サイクル読出および 3 サイクル書込オペレーションを示すタイミングダイアグラムの図である。

【図 5 1】 4 つの 2 サイクル読出オペレーションを示すタイミングダイアグラムの図である。

【図 5 2】 8 つの 2 サイクル DPIX オペレーションを示すタイミングダイアグラムの図である。

【図 5 3】 図 5 2 の 8 つの 2 サイクル DPIX オペレーションを続けて示す図である。

【図 5 4】 いくつかのレジスタのためのデータフォーマットを示す図である。

【図 5 5】 PixelConfig レジスタのためのデータフォーマットを示す図である。

【図 5 6】 StencilDepthConfig レジスタのためのデータフォーマットを示す図である。

【図 5 7】 ColorOp[0] レジスタのためのデータフォーマットを示す図である。

【図 5 8】 ColorOp[1] レジスタのためのデータフォーマットを示す図である。

【図 5 9】 ConstantColor レジスタのためのデータフォーマットを示す図である。

【図 6 0】 DisplayConfig レジスタのためのデータフォーマットを示す図である。

【図 6 1】 WREG オペレーションから ByteMask および ML オペレーションによりどのように高速フィルを行なうかを示すタイミングダイアグラムの図である。

【図 6 2】 FL オペレーションによってどのように非常に高速のフィルを行なうかを示すタイミングダイアグラムの図である。

【図 6 3】 入力データフォーマットを示すブロック図である。

【図 6 4】 入力データフォーマットの入力ルーティング層を示す図である。

【図 6 5】 入力データフォーマットを示す図である。

【図 6 6】 色およびデプスアキュムレータを示すブロック図である。

【図 6 7】 ROP/ブレンドユニットのための色成分フォーマットを示す図である。

【図 6 8】 出力データフォーマットを示すブロック図である。

【図 6 9】 RDAT、RPIX オペレーションの出力データルーティングを示す図である。

【図 7 0】 出力データルーティングを示す図である。

【図 7 1】 RPIX オペレーションの出力データルーティングを示す図である。

【図 7 2】 出力データルーティングを示す図である。

【図 7 3】 出力データルーティングを示す図である。

【図 7 4】 出力データルーティングを示す図である。

【図 7 5】 9 ビット/ピクセルの出力データルーティングを示す図である。

【図 7 6】 出力データルーティングを示す図である。

【図 7 7】 出力データルーティングを示す図である。

【図 7 8】 出力データルーティングを示す図である。

【図 7 9】 出力データルーティングを示す図である。

【図 8 0】 出力データルーティングを示す図である。

【図 8 1】 出力データルーティングを示す図である。

【図 8 2】 出力データルーティングを示す図である。

【図 8 3】 出力データルーティングを示す図である。

【図 8 4】 SRAM-ピクセル ALU フォーマットを示すブ



ック図である。

【図 85】 8、16、32ビット/ピクセルのSRAMピクセルパッパ編成を示す図である。

【図 86】 64ビット/ピクセルのSRAMピクセルパッパ編成を示す図である。

【図 87】 96ビット/ピクセルのSRAMピクセルパッパ編成を示す図である。

【図 88】 128ビット/ピクセルのSRAMピクセルパッパ編成を示す図である。

【図 89】 アンパック関数を示す図である。

【図 90】 UnpackDepth関数を示す図である。

【図 91】 UnpackStencil関数を示す図である。

【図 92】 UnpackWidth関数を示す図である。

【図 93】 ピクセルALU-SRAMフォーマットを示すブロック図である。

【図 94】 PackColor関数を示す図である。

【図 95】 PackDepth関数を示す図である。

【図 96】 PackExtra関数を示す図である。

【図 97】 ピクセルALU-SRAMマスク生成を示す図である。

【図 98】 8ビット/ピクセルディスプレイマッピングを示す図である。

【図 99】 RPIX、SPIXおよびDPPIXオペレーションのための32ビット/ピクセルキャッシュライン編成を示す図である。

【図 100】 RDATAオペレーションを用いる8ビット/ピクセルキャッシュライン編成を示す図である。

【図 101】 8ビット/ピクセルフォーマットを示す図である。

【図 102】 8ビット/ピクセル表示リフレッシュを示す図である。

【図 103】 16ビット/ピクセルディスプレイマッピングを示す図である。

【図 104】 RPIX、SPIXおよびDPPIXオペレーションのための16ビット/ピクセルキャッシュライン編成を示す図である。

【図 105】 RDATAオペレーションを用いる16ビット/ピクセルキャッシュライン編成を示す図である。

【図 106】 4:4:4:4の16ビット/ピクセルフォーマットを示す図である。

【図 107】 4:4:4:4の16ビット/ピクセル表示リフレッシュを示す図である。

【図 108】 5:6:5の16ビット/ピクセルフォーマットを示す図である。

【図 109】 5:6:5の16ビット/ピクセル表示リフレッシュを示す図である。

【図 110】 1:5:5:5の16ビット/ピクセルフォーマットを示す図である。

【図 111】 1:5:5:5の16ビット/ピクセル表示リフレッシュを示す図である。

10

20

30

40

50

【図 112】 32ビット/ピクセルディスプレイマッピングを示す図である。

【図 113】 RPIX、SPIXおよびDPPIXオペレーションのための32ビット/ピクセルキャッシュライン編成を示す図である。

【図 114】 RDATAオペレーションを用いる32ビット/ピクセルキャッシュライン編成を示す図である。

【図 115】 8:8:8:8の32ビット/ピクセルフォーマットを示す図である。

【図 116】 8:8:8:8の32ビット/ピクセル表示リフレッシュを示す図である。

【図 117】 10:10:10:10の32ビット/ピクセルフォーマットを示す図である。

【図 118】 2:10:10:10の32ビット/ピクセル表示リフレッシュを示す図である。

【図 119】 64ビット/ピクセルディスプレイマッピングを示す図である。

【図 120】 RPIX、SPIXおよびDPPIXオペレーションのための64ビット/ピクセルキャッシュライン編成を示す図である。

【図 121】 RDATAオペレーションを用いる64ビット/ピクセルキャッシュライン編成を示す図である。

【図 122】 4:8:8:8の64ビット/ピクセルフォーマットを示す図である。

【図 123】 4:8:8:8の64ビット/ピクセル表示リフレッシュを示す図である。

【図 124】 2:10:10:10の64ビット/ピクセルフォーマットを示す図である。

【図 125】 4:10:10:10の64ビット/ピクセル表示リフレッシュを示す図である。

【図 126】 4:28:2\* (4:4:4:4) の64ビット/ピクセルフォーマットを示す図である。

【図 127】 4:28:2\* (4:4:4:4) の64ビット/ピクセル表示リフレッシュを示す図である。

【図 128】 4:4:24:2\* (4:4:4:4) の64ビット/ピクセルフォーマットを示す図である。

【図 129】 4:4:24:2\* (4:4:4:4) の64ビット/ピクセル表示リフレッシュを示す図である。

【図 130】 4:28:2\* (5:6:5) の64ビット/ピクセルフォーマットを示す図である。

【図 131】 4:28:2\* (5:6:5) の64ビット/ピクセル表示リフレッシュを示す図である。

【図 132】 4:4:24:2\* (5:6:5) の64ビット/ピクセルフォーマットを示す図である。

【図 133】 4:4:24:2\* (5:6:5) の64ビット/ピクセル表示リフレッシュを示す図である。

【図 134】 4:28:2\* (1:5:5:5) の64ビット/ピクセルフォーマットを示す図である。

【図 135】 4:28:2\* (1:5:5:5) の64ビット/ピクセル表示リフレッシュを示す図である。

【図 1 3 6】4:4:24:2\* (1:5:5:5) の 6 4 ビット/ピクセルフォーマットを示す図である。

【図 1 3 7】4:4:24:2\* (1:5:5:5) の 6 4 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 3 8】9 6 ビット/ピクセルディスプレイマッピングを示す図である。

【図 1 3 9】RPIX、SPIXおよびDPPIXオペレーションのための 9 6 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 4 0】RDATオペレーションを用いる 9 6 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 4 1】4:28:2\* (8:8:8:8) の 9 6 ビット/ピクセルフォーマットを示す図である。

【図 1 4 2】4:28:2\* (8:8:8:8) の 9 6 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 4 3】4:28:2\* (8:8:8:8) の 9 6 ビット/ピクセルフォーマットを示す図である。

【図 1 4 4】4:28:2\* (8:8:8:8) の 9 6 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 4 5】4:4:24:2\* (8:8:8:8) の 9 6 ビット/ピクセルフォーマットを示す図である。

【図 1 4 6】4:4:24:2\* (8:8:8:8) の 9 6 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 4 7】4:28:2\* (2:10:10:10) の 9 6 ビット/ピクセルフォーマットを示す図である。

【図 1 4 8】4:28:2\* (2:10:10:10) の 9 6 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 4 9】4:4:24:2\* (10:10:10) の 9 6 ビット/ピクセルフォーマットを示す図である。

【図 1 5 0】4:4:24:2\* (10:10:10) の 9 6 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 5 1】4:4:24:4\* (4:4:4:4) の 9 6 ビット/ピクセルフォーマットを示す図である。

【図 1 5 2】4:4:24:4\* (4:4:4:4) の 9 6 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 5 3】1 2 8 ビット/ピクセルディスプレイマッピングを示す図である。

【図 1 5 4】RPIX、SPIXおよびDPPIXオペレーションのための 1 2 8 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 5 5】RDATオペレーションを用いる 1 2 8 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 5 6】8:32:8:2\* (8:8:8:8:8) の 1 2 8 ビット/ピクセルフォーマットを示す図である。

【図 1 5 7】8:32:8:2\* (8:8:8:8:8) の 1 2 8 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 5 8】8:32:8:2\* (8:2:10:10:10) の 1 2 8 ビット/ピクセルフォーマットを示す図である。

【図 1 5 9】8:32:8:2\* (8:10:10:10) の 1 2 8 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 6 0】8:8:32:2\* (10:10:10:10) の 1 2 8 ビット/ピクセルフォーマットを示す図である。

【図 1 6 1】8:8:32:2\* (10:10:10:10) の 1 2 8 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 6 2】2 つのピクセルと各ピクセル内の各サンプルの位置を示す図である。

【図 1 6 3】ピクセル内のサンプル間のオフセットを示す図である。

【図 1 6 4】2 5 6 ビット/ピクセルディスプレイマッピングを示す図である。

【図 1 6 5】RPIX、SPIXおよびDPPIXオペレーションのための 2 5 6 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 6 6】RDATオペレーションを用いる 2 5 6 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 6 7】8:2\* (8:8:8:8:8) :6\* (8:8:8:8:32) の 2 5 6 ビット/ピクセルフォーマットを示す図である。

【図 1 6 8】8:2\* (8:8:8:8:8) :6\* (8:8:8:8:32) の 2 5 6 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 6 9】SRAMピクセルバッファ読出/書込フォーマットを示す図である。

【図 1 7 0】5 1 2 ビット/ピクセルディスプレイマッピングを示す図である。

【図 1 7 1】RPIX、SPIXおよびDPPIXオペレーションのための 5 1 2 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 7 2】RDATオペレーションを用いる 5 1 2 ビット/ピクセルキャッシュライン編成を示す図である。

【図 1 7 3】8:2\* (8:8:8:8:8) :6\* (8:8:8:8:32) の 5 1 2 ビット/ピクセルフォーマットを示す図である。

【図 1 7 4】8:2\* (8:8:8:8:8) :6\* (8:8:8:8:32) の 5 1 2 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 7 5】SRAMピクセルバッファ読出/書込フォーマットを示す図である。

【図 1 7 6】8:2\* (8:10:10:10) :6\* (10:10:10:32) の 5 1 2 ビット/ピクセルフォーマットを示す図である。

【図 1 7 7】8:2\* (8:2:10:10:10) :6\* (2:10:10:10:32) の 5 1 2 ビット/ピクセル表示リフレッシュを示す図である。

【図 1 7 8】SRAMピクセルバッファ読出/書込フォーマットを示す図である。

【図 1 7 9】I/Oバスにわたっての高速同時双方向送受信をサポートするデュアルピクセル 3 DRAMチップの代替のアーキテクチャを示すブロック図である。

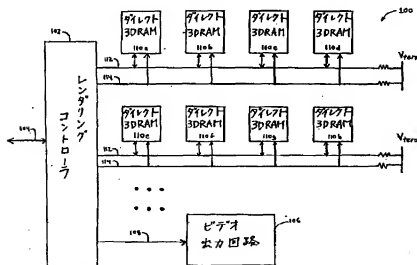
【図 1 8 0】I/Oバスにわたっての高速同時双方向送受信をサポートするデュアルピクセル 3 DRAMチップのための第 2 の代替のアーキテクチャを示すブロック図である。

## 【符号の説明】

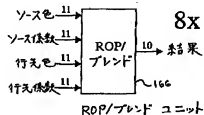
100 グラフィックスサブシステム、102 レンダリングコントローラ、104 インタフェース、106 ビデオ出力回路、108 ビデオ出力チャネル、110 デュアルビクセル3DRAMチップ、112 レンダリングバス、114アドレスおよび制御バス、116 DRAMアレイ、118 SRAMビクセルバッファ、120、121 ビクセル算術論理演算装置 (ALU)、122 グローバルバス、123 グローバル書込バス、124 I/Oバス、125 グローバル読出バス、126 デマルチプレクサ、127 バイプラインレジスタ、128 入力データバス、129、131 入力データバス、130 入力データフォーマッタ、132 SRAM出力データバス、134 出力データフォーマッタ、135 出力データバス、136 出力データマルチプレクサ、137 バイプラインレジスタ、138、139 データバス、140 ビクセルALUからSRAMへのフォーマッタ、141 データバス、142、143 SRAMビクセルバッファデータバス、145 書込マスクバイプラインレジスタ、146 アドレスおよび制御入力バス、1\*20

\*47 書込マスクバス、148 アドレスおよび制御デマルチプレクサ、150 アドレスおよび制御バス、151 デコーダ、152 ビクセルALUオペレーションチャネル、154 グローバルバスオペレーションチャネル、156 バンクオペレーションチャネル、158 DRAMバンク、160センス増幅器、166 ROP/ブレンドユニット、168 デプスユニット、170 ステンシルユニット、172 ウィンドウIDユニット、174 ROPユニット、176 ブレンドユニット、178 8ビットブレンドユニット、18010ビットブレンドユニット、182 ROPレジスタ、184 バターンレジスタ、186 デザ計算装置、188、190 乗算器、192 加算器、194 切捨て装置、196 クランプ装置、210 16ビットマスクレジスタ、216 マスクレジスタ、218 基準レジスタ、220 ファンクションレジスタ、224 キャッシュライン、226 ダーティ・タグSRAM、230 キャッシュラインのバンクおよびコラムタグ、231 バンクバイプラインレジスタ、232 コラムバイプラインレジスタ。

【図1】



【図7】

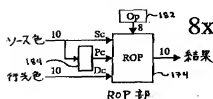


【図30】

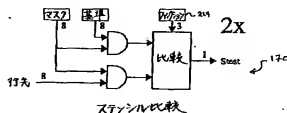


タイトルコメント

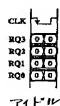
【図8】



【図16】



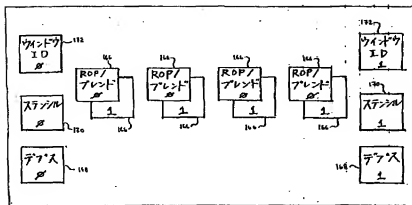
【図38】



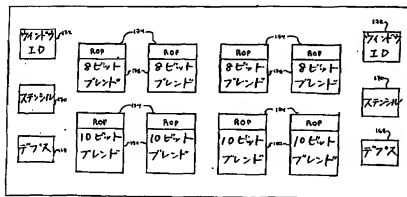
タイトル



【図4】



【図5】



【図10】

入力	00000000	1.511
値	00000000000000000000000000000000	1.261,121
デシマ(8)	00000000000000000000000000000000	-480.480
デシマ(6)	00000000000000000000000000000000	-1920.1920
デシマ(5)	00000000000000000000000000000000	-3840.3840
デシマ(4)	00000000000000000000000000000000	-7680.7680
ゼロ	00000000000000000000000000000000	-268,801.529,922
ゼロ	00000000000000000000000000000000	-263.517
ゼロ	00000000000000000000000000000000	0.255

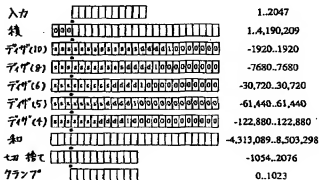
8ビットブレンドユニット計算

【図11】

		X%4				X%4			
		0	1	2	3	0	1	2	3
Y%4	0	-15 32	+1 32	-11 32	+5 32	10001	00001	10101	00101
	1	+9 32	-7 32	+13 32	-3 32	01001	11001	01101	11101
	2	-9 32	+7 32	-13 32	+3 32	10111	00111	10011	00011
	3	+15 32	-1 32	+11 32	-5 32	01111	11111	01011	11011

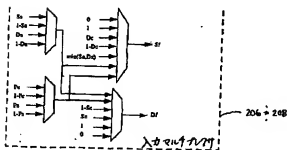
ドット分数量組織のデシマログスル

【图 13】

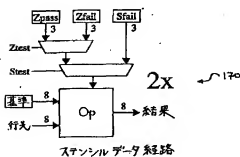


10ビットブレンドユニットの計算

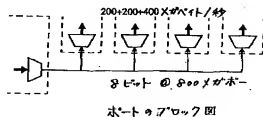
【图 15】



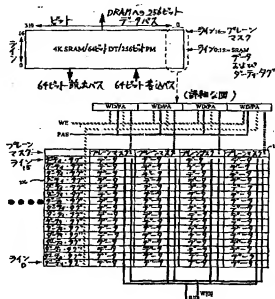
【图 18】



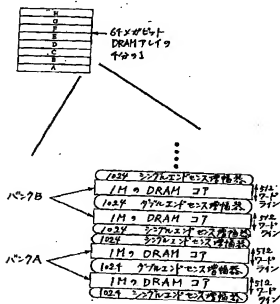
【图 24】



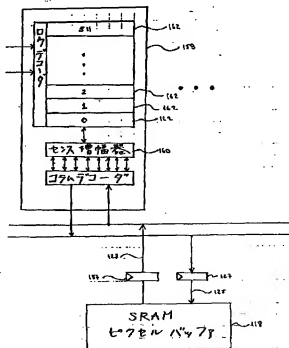
【圖 20】



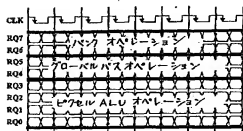
【圖 2 2】



【图 2 3】

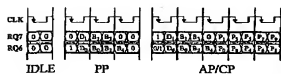


【图 25】



アドレスおよび制御  
プロトコル

【図 26】



バンクオペレーション  
フロッタコレ

【図 27】



ページ "リチャージ"

【図 42】



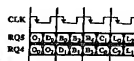
データ転送

【図 28】



ページ アクセス

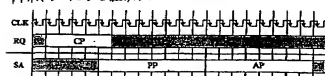
【図 31】



RLコマンド、  
WLコマンド、  
MLコマンド、  
および  
CLコマンド

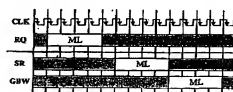
【図 29】

CP(ページ変更)は、APオペレーションが終了  
PPオペレーションと組み合わせて



ページ変更

【図 35】



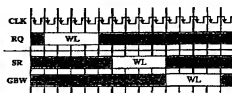
ML オペレーション

【図 33】



RL オペレーション

【図 34】



WL オペレーション

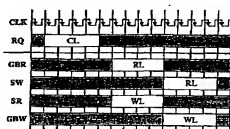


【図36】



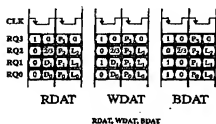
FL オペレーション

【図37】



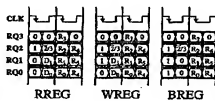
CL オペレーション

【図39】



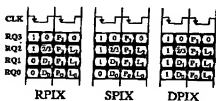
RDAT, WDAT, BDAT

【図40】



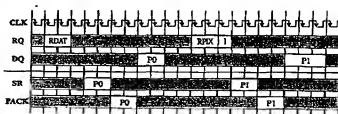
RREG, WREG, BREG

【図41】



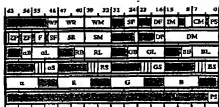
RPIX, SPIX, DPIX

【図43】

2サイクル RDAT オペレーション  
続いて3サイクル RPIX オペレーション

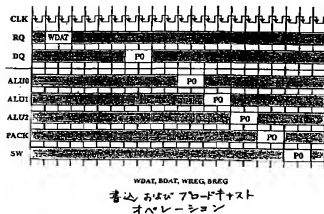
【図54】

Pixel/Config  
Stencil/Depth/Config  
ColorOp[0]  
ColorOp[1]  
ConstantColor  
Display/Config

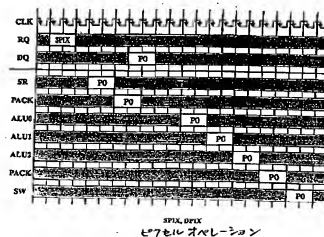


レジスタ データフォーマット

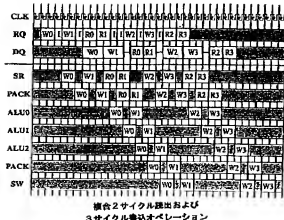
【図44】



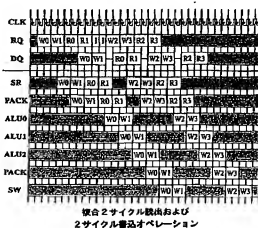
【図45】



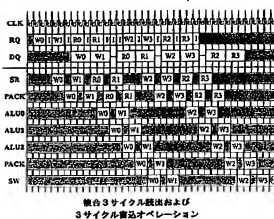
【図48】



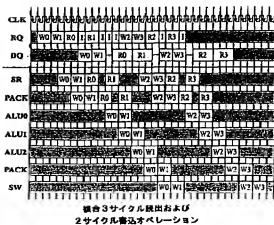
【図47】



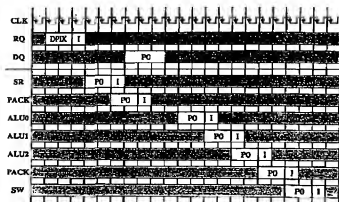
【図50】



【図49】

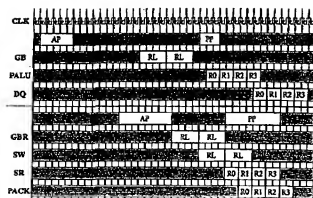


【図46】



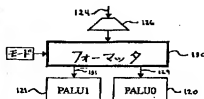
32ビットのDPIX転送

【図51】

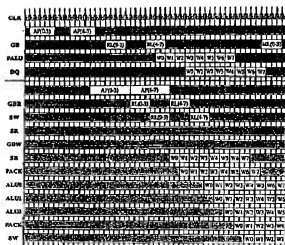


4つの2サイクル読出しオペレーション

【図63】

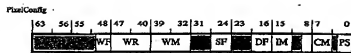
DQからピクセルALUへの  
ルーティング

【図52】



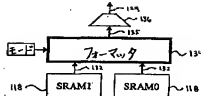
8つの2サイクルDPIXオペレーション

【図55】



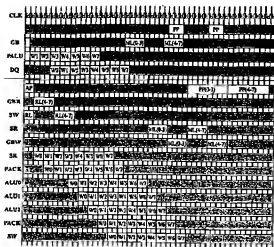
PixelConfig レジスタのデータフォーマット

【図68】

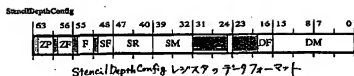


SRAMからDQへのルーティング

【図 53】

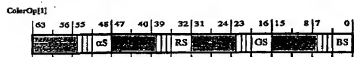
8ビットSTY DP1Xオペレーション  
(続き)

【図 56】



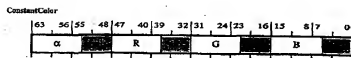
Stencil/DepthConfig レジスタフォーマット

【図 58】



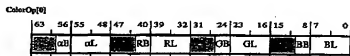
ColorOp[1] レジスタフォーマット

【図 59】



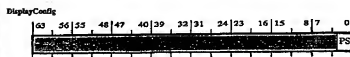
ConstantColor レジスタフォーマット

【図 57】



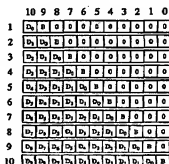
ColorOp[0] レジスタフォーマット

【図 60】

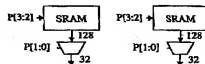


DisplayConfig レジスタフォーマット

【図 67】

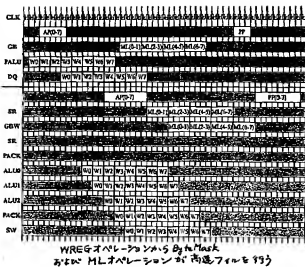
ROP/Blend ユニットの構成  
288bit フォーマット

【図 70】

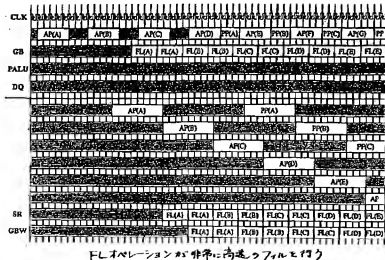


SRAM 288bit フォーマット

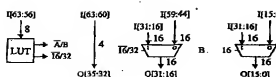
【図61】



【図62】

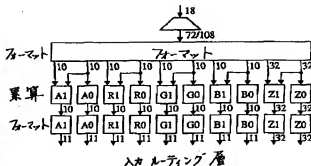


【図73】

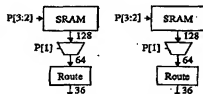


SRAMからDQへのルータリング

【図64】

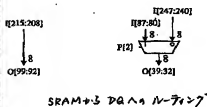


【図72】



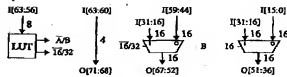
SRAMからDQへのルータリング

【図77】



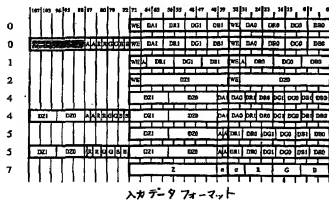
SRAMからDQへのルータリング

【図74】

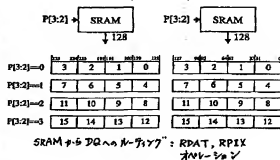


SRAMからDQへのルータリング

【図 65】

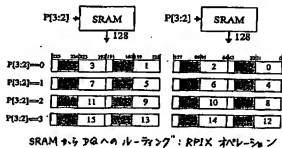
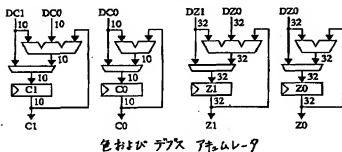


【図 69】

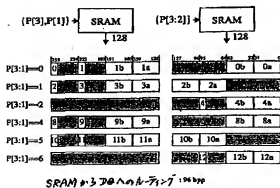


【図 71】

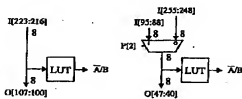
【図 66】



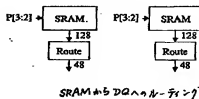
【図 75】



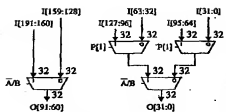
【図 76】



【図 80】

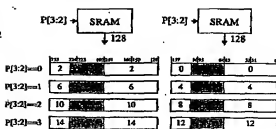


【図78】



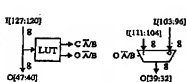
SRAM から DQ へのルーティング

【図79】



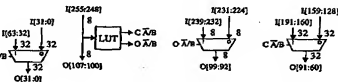
SRAM から DQ へのルーティング

【図81】



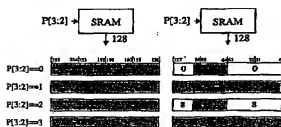
SRAM から DQ へのルーティング

【図82】



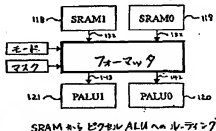
SRAM から DQ へのルーティング

【図83】



SRAM から DQ へのルーティング

【図84】



SRAM から ビックセル ALU へのルーティング

【図98】

【図85】

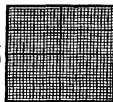
3a	2c	1c	0c
7c	6c	5c	4c
11c	10c	9c	8c
15c	14c	13c	12c

SRAM 構成 8、16、32 bpp

【図86】

3d	2b	1d	1b
7d	7b	5d	5b
11d	11b	9d	9b
15d	15b	13d	13b

SRAM 構成 64 bpp



8 bpp 74シリーズルーティング

【図 87】

0dc	1dc	10c	11c
2dc	3dc	30c	31c
8dc	9dc	90c	91c
10dc	11dc	110c	111c

SRAM 構成 96 bpp

【図 88】

2b	2a	0b	0a
4b	4a	6b	6a
10b	10a	8b	8a
12b	12a	12a	12a

SRAM 構成 128 bpp

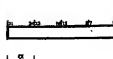
【図 101】



8 bpp ピクセルフォーマット

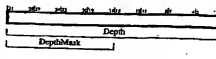
【図 103】

【図 89】



Unpack 関数

【図 90】



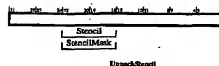
UnpackDepth



16 bpp ディスクリプティング

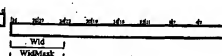
【図 100】

【図 91】



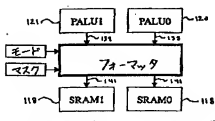
UnpackStencil

【図 92】



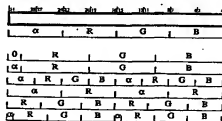
UnpackWid

【図 93】



ピクセル ALU から SRAM への変換

【図 94】



PackColor

RDAT を用いる

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

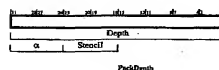
8 bpp カラーライン構成

8 bpp カラーライン構成

8 bpp カラーライン構成

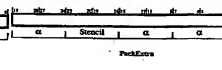
8 bpp カラーライン構成

【図 95】



PackDepth

【図 96】



PackExtra

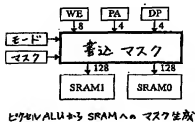
【図 106】



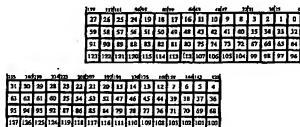
16 bpp ディスクリプティング



【图 9-7】



【图 9-9】

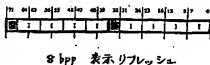


【图 112】

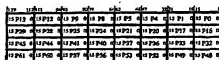


RPIX、SPHX、DPIX のための 8bpp マッシュアップライン生成

【图 102】



【圖 104】

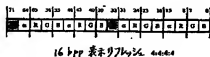


【圖 105】



RPIX、SPIX、DPIXのためのキャッシュライン構成

【圖 107】



【图 108】



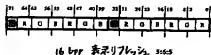
【图 110】



【111】



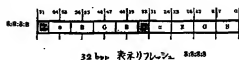
【图 109】

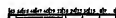
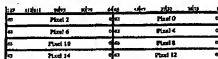
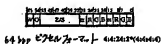
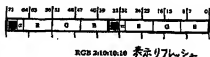
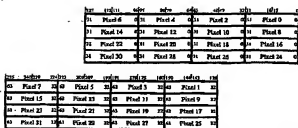


【图 1 1 5】



【图 116】

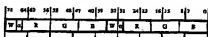
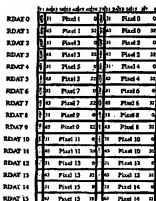
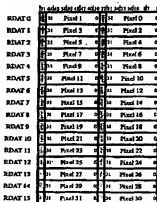




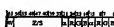
64 bppピクセルフォーマット 2:10:10:10



64 bpp 6791672-241 42224565



64bpp 表示リフレッシュ 4:10:10:10



64bppピクセルフォーマット 4:2:2 (4:4:4)



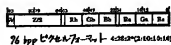
【図144】



【図146】



【図147】



【図148】

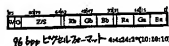


【図153】

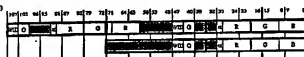


128bpp ディスプレイマッピング

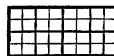
【図149】



【図150】

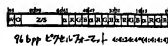


【図164】



256bpp ディスプレイマッピング

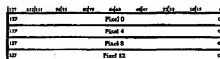
【図151】



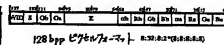
【図152】



【図154】



【図156】



【図157】



RPix, SPix, DPixのための、128bppフォーマット定義

【図155】

RDAT 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RDAT 1	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62
RDAT 2	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94
RDAT 3	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126
RDAT 4	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158
RDAT 5	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190
RDAT 6	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222
RDAT 7	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254
RDAT 8	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286
RDAT 9	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318
RDAT 10	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350
RDAT 11	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382
RDAT 12	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414
RDAT 13	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446
RDAT 14	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478
RDAT 15	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510

RDAT を用い 3

128 bpp カラーライン構成

【図158】

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

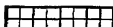
128 bpp ビグエルフォーマット 8.32.8.2\*(8.3.10.10.10)

【図160】

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

128 bpp ビグエルフォーマット 8.32.8.2\*(8.3.10.10.10)

【図170】



512 bpp テキスチャリング

【図161】

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

128 bpp 表示リフレッシュ 8.32.8.2\*(8.3.10.10.10)

【図162】

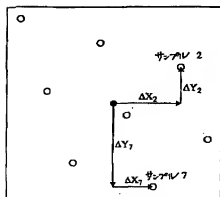


【図159】

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

128 bpp 表示リフレッシュ 8.32.8.2\*(8.3.10.10.10)

【図163】



【図165】

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

RPIX、SPIX、DPIX のための、256 bpp カラーライン構成

【図168】

【図167】

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

256 bpp ビグエルフォーマット

128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
W2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

256 bpp 表示リフレッシュ 8.32.8.2\*(8.3.10.10.10.10.10.10)

【図166】

RDAT 0	10	Pixel 0	120	21	Pixel 0	4
RDAT 1	10	Pixel 0	120	21	Pixel 0	4
RDAT 2	10	Pixel 0	120	21	Pixel 0	4
RDAT 3	10	Pixel 0	120	21	Pixel 0	4
RDAT 4	10	Pixel 0	120	21	Pixel 0	4
RDAT 5	10	Pixel 0	120	21	Pixel 0	4
RDAT 6	10	Pixel 0	120	21	Pixel 0	4
RDAT 7	10	Pixel 0	120	21	Pixel 0	4
RDAT 8	10	Pixel 0	120	21	Pixel 0	4
RDAT 9	10	Pixel 0	120	21	Pixel 0	4
RDAT 10	10	Pixel 0	120	21	Pixel 0	4
RDAT 11	10	Pixel 0	120	21	Pixel 0	4
RDAT 12	10	Pixel 0	120	21	Pixel 0	4
RDAT 13	10	Pixel 0	120	21	Pixel 0	4
RDAT 14	10	Pixel 0	120	21	Pixel 0	4
RDAT 15	10	Pixel 0	120	21	Pixel 0	4

RDAT を用いる

256 bpp カラーライン構成

【図169】

251	at	R1	G1	B1	250	at	R0	G0	B0
253	at	R3	G3	B3	252	at	R2	G2	B2
255	at	R5	G5	B5	254	at	R4	G4	B4

SRAM 読出/書き込み

【図172】

RDAT 0	10	Pixel 0	120	21	Pixel 0	4
RDAT 1	10	Pixel 0	120	21	Pixel 0	4
RDAT 2	10	Pixel 0	120	21	Pixel 0	4
RDAT 3	10	Pixel 0	120	21	Pixel 0	4
RDAT 4	10	Pixel 0	120	21	Pixel 0	4
RDAT 5	10	Pixel 0	120	21	Pixel 0	4
RDAT 6	10	Pixel 0	120	21	Pixel 0	4
RDAT 7	10	Pixel 0	120	21	Pixel 0	4
RDAT 8	10	Pixel 0	120	21	Pixel 0	4
RDAT 9	10	Pixel 0	120	21	Pixel 0	4
RDAT 10	10	Pixel 0	120	21	Pixel 0	4
RDAT 11	10	Pixel 0	120	21	Pixel 0	4
RDAT 12	10	Pixel 0	120	21	Pixel 0	4
RDAT 13	10	Pixel 0	120	21	Pixel 0	4
RDAT 14	10	Pixel 0	120	21	Pixel 0	4
RDAT 15	10	Pixel 0	120	21	Pixel 0	4

RDAT を用いる

512 bpp カラーライン構成

【図171】

10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4

RPIX、SPIX、DPPIX の読み、512 bpp カラーライン構成

【図173】

251	at	R1	G1	B1	250	at	R0	G0	B0
253	at	R3	G3	B3	252	at	R2	G2	B2
255	at	R5	G5	B5	254	at	R4	G4	B4

512 bpp ビタブルフォーマット (256bpp/256bpp/256bpp/256bpp)

【図174】

10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4
10	Pixel 0	120	21	Pixel 0	4

512 bpp 表示リフレクタ (256bpp/256bpp/256bpp/256bpp)

【図175】

251	at	R1	G1	B1	250	at	R0	G0	B0
253	at	R3	G3	B3	252	at	R2	G2	B2
255	at	R5	G5	B5	254	at	R4	G4	B4

SRAM 読出/書き込み



フロントページの続き

- |             |                       |               |                       |
|-------------|-----------------------|---------------|-----------------------|
| (31)優先権主張番号 | 09/264261             | * (31)優先権主張番号 | 09/264281             |
| (32)優先日     | 平成11年3月8日(1999. 3. 8) | (32)優先日       | 平成11年3月8日(1999. 3. 8) |
| (33)優先権主張国  | 米国 (US)               | * (33)優先権主張国  | 米国 (US)               |
- (54)【発明の名称】 記憶装置、データフォーマッタ、データにアクセスする方法、データの領域をクリアする方法、データを圧縮する方法、データをフォーマット化する方法、グラフィックシステムおよびグラフィックシステムを動作させる方法